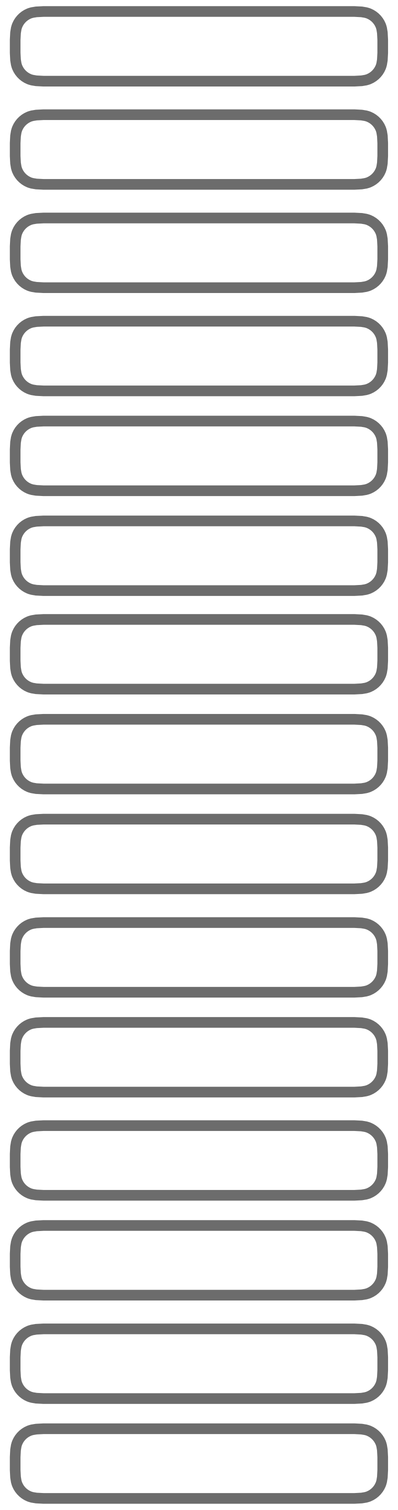
# Efficient Transformers

Angela Fan

# Standard Neural Networks

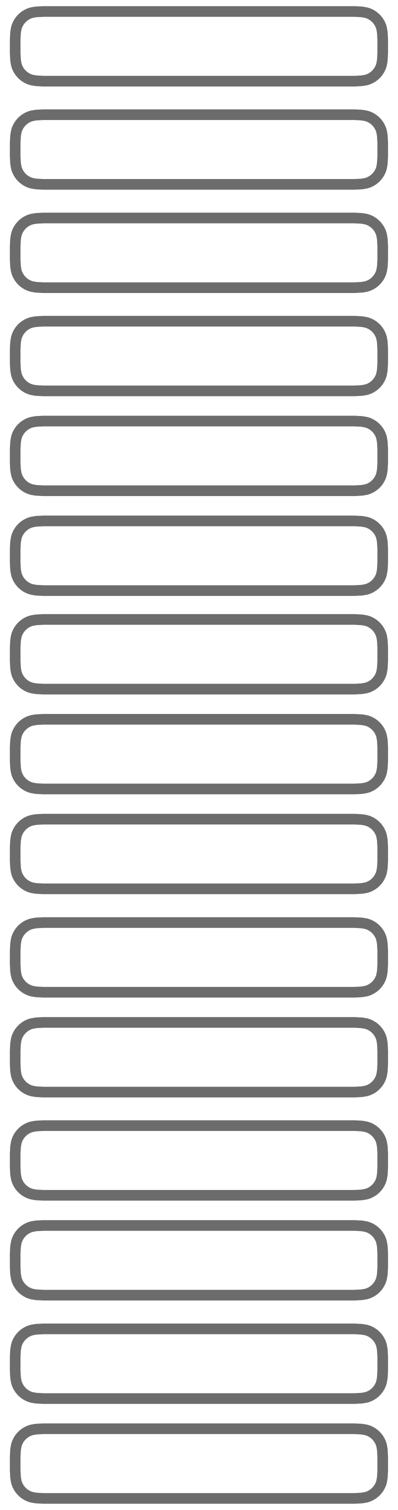
TRAINING TIME



- Overparameterized

# Standard Neural Networks

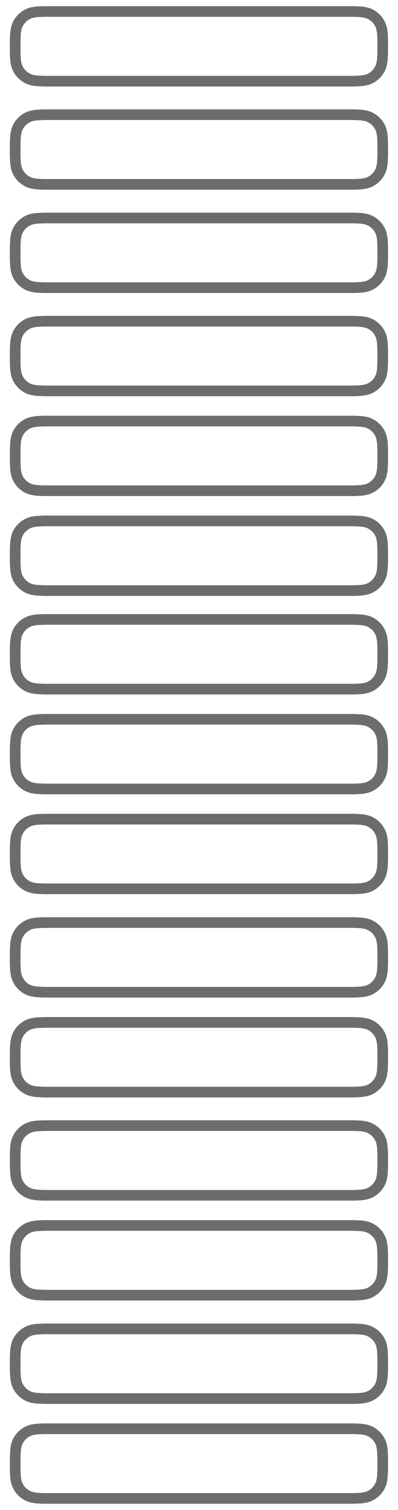
TRAINING TIME



- Overparameterized
- Redundant

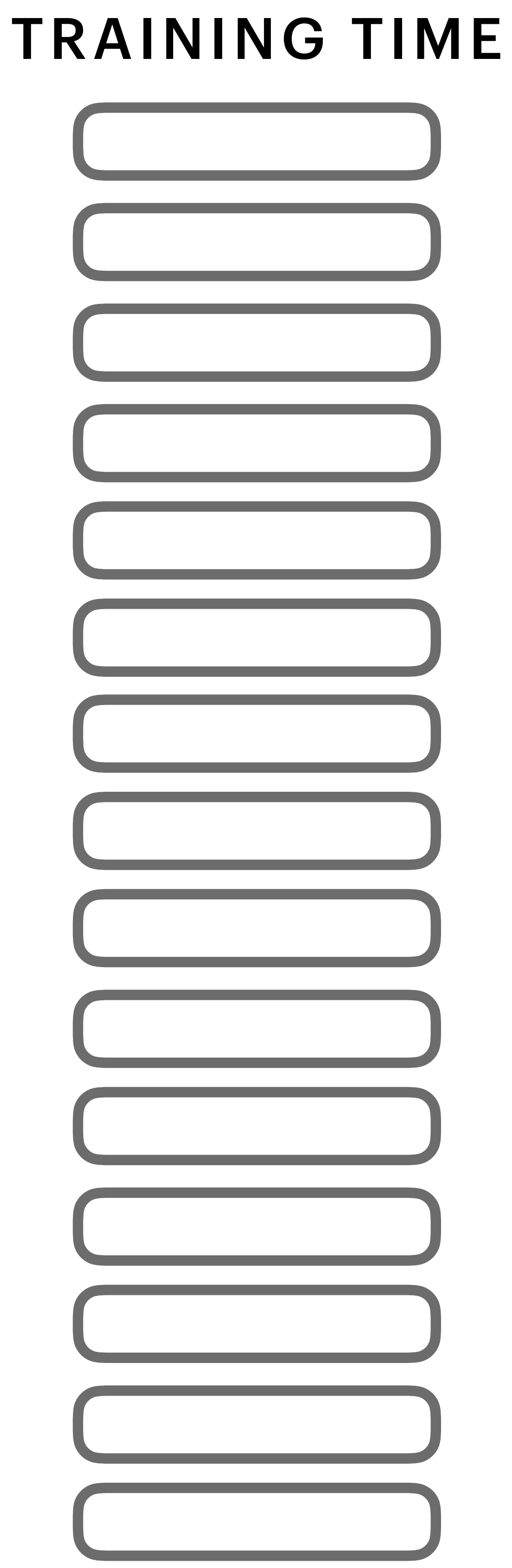
# Standard Neural Networks

TRAINING TIME



- Overparameterized
- Redundant
- Overfitting

# Standard Neural Networks




- Overparameterized
- Redundant
- Overfitting
- Too Large for Practical Applications

# Techniques for Smaller Networks

- Train Smaller Network from Scratch

# Techniques for Smaller Networks

- Train Smaller Network from Scratch
- Sparsity Inducing Training

# Techniques for Smaller Networks

- Train Smaller Network from Scratch
- Sparsity Inducing Training
- Knowledge Distillation

# Techniques for Smaller Networks

- Train Smaller Network from Scratch
- Sparsity Inducing Training
- Knowledge Distillation
- Pruning
- Weight Sharing

# Techniques for Smaller Networks

- Train Smaller Network from Scratch
- Sparsity Inducing Training
- Knowledge Distillation
- Pruning
- Weight Sharing
- Quantization

# Techniques for Smaller Networks

- Train Smaller Network from Scratch
- Sparsity Inducing Training
- Knowledge Distillation
- Pruning
- Weight Sharing
- Quantization
- More efficient architectures

Caveat: this is a **brief** overview focused on **Transformers**

# What to think about when talking about efficiency?

- Training Time
- Inference Time
- Model size
- Energy

# What to think about when talking about efficiency?

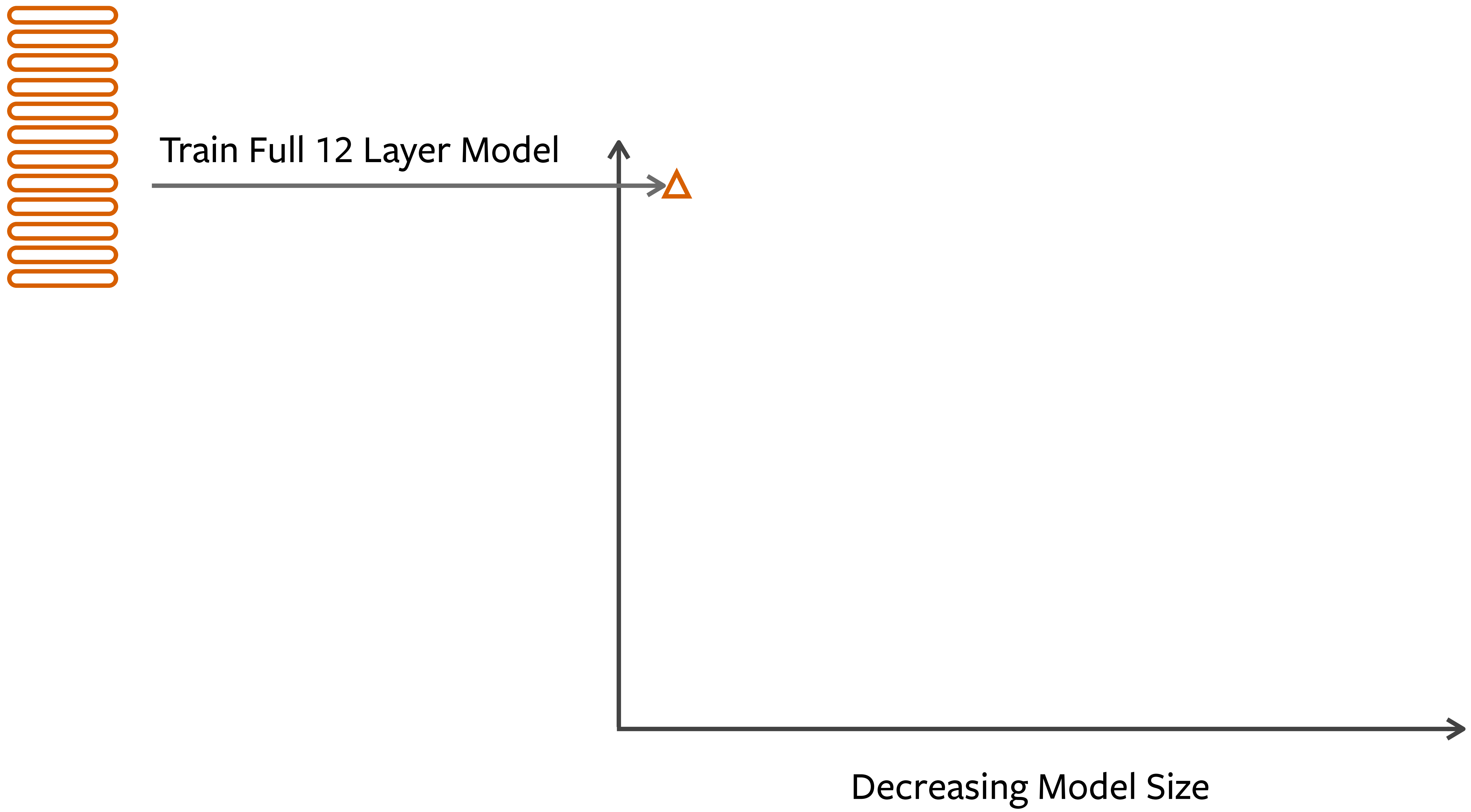
- Training Time
- Inference Time
- Model size
- Energy

not all techniques improve all of these areas

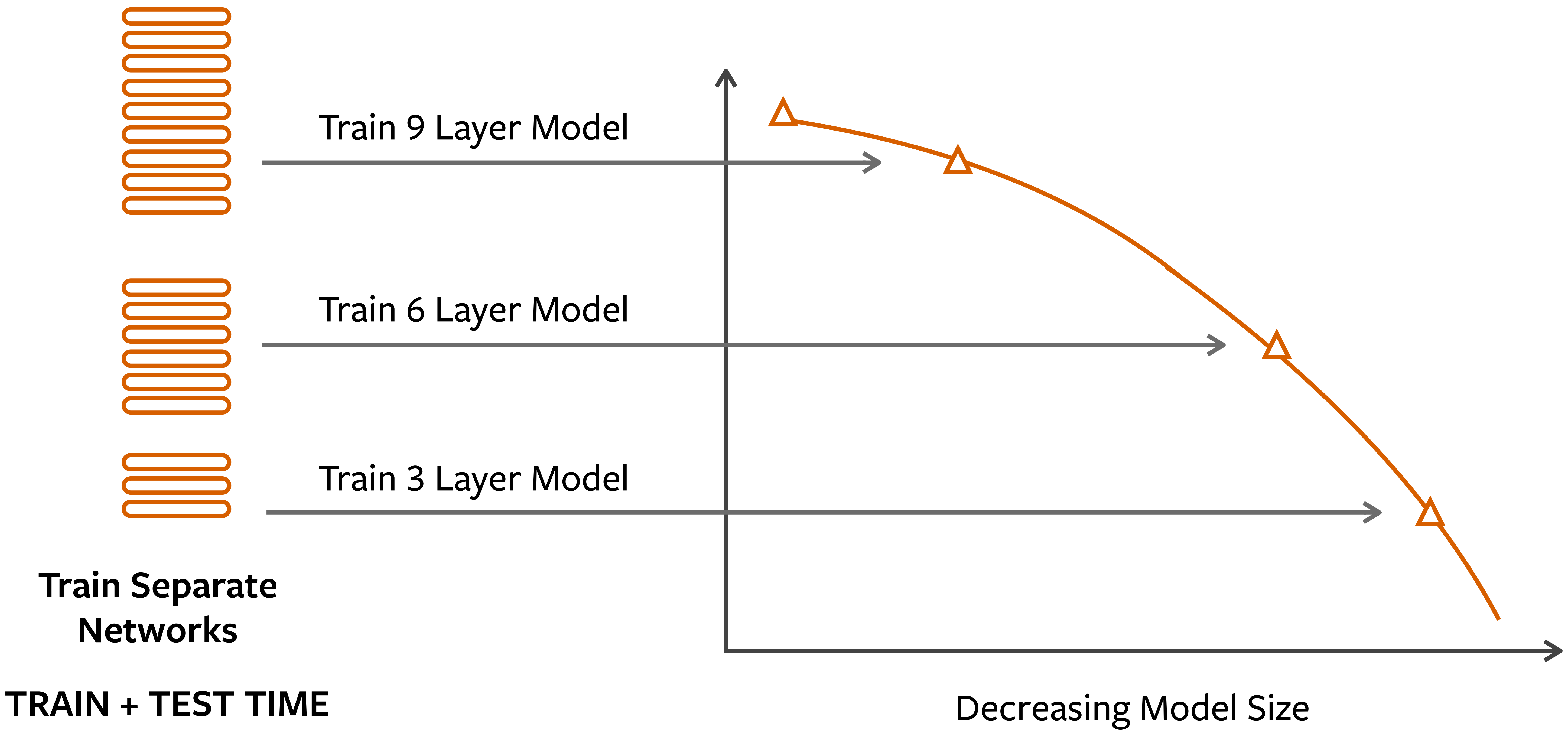
# Techniques for Smaller Networks

- **Train Smaller Network from Scratch**
- Sparsity Inducing Training
- Knowledge Distillation
- Pruning
- Weight Sharing
- Quantization
- More efficient architectures

# Training a Smaller Model from Scratch



# Training a Smaller Model from Scratch



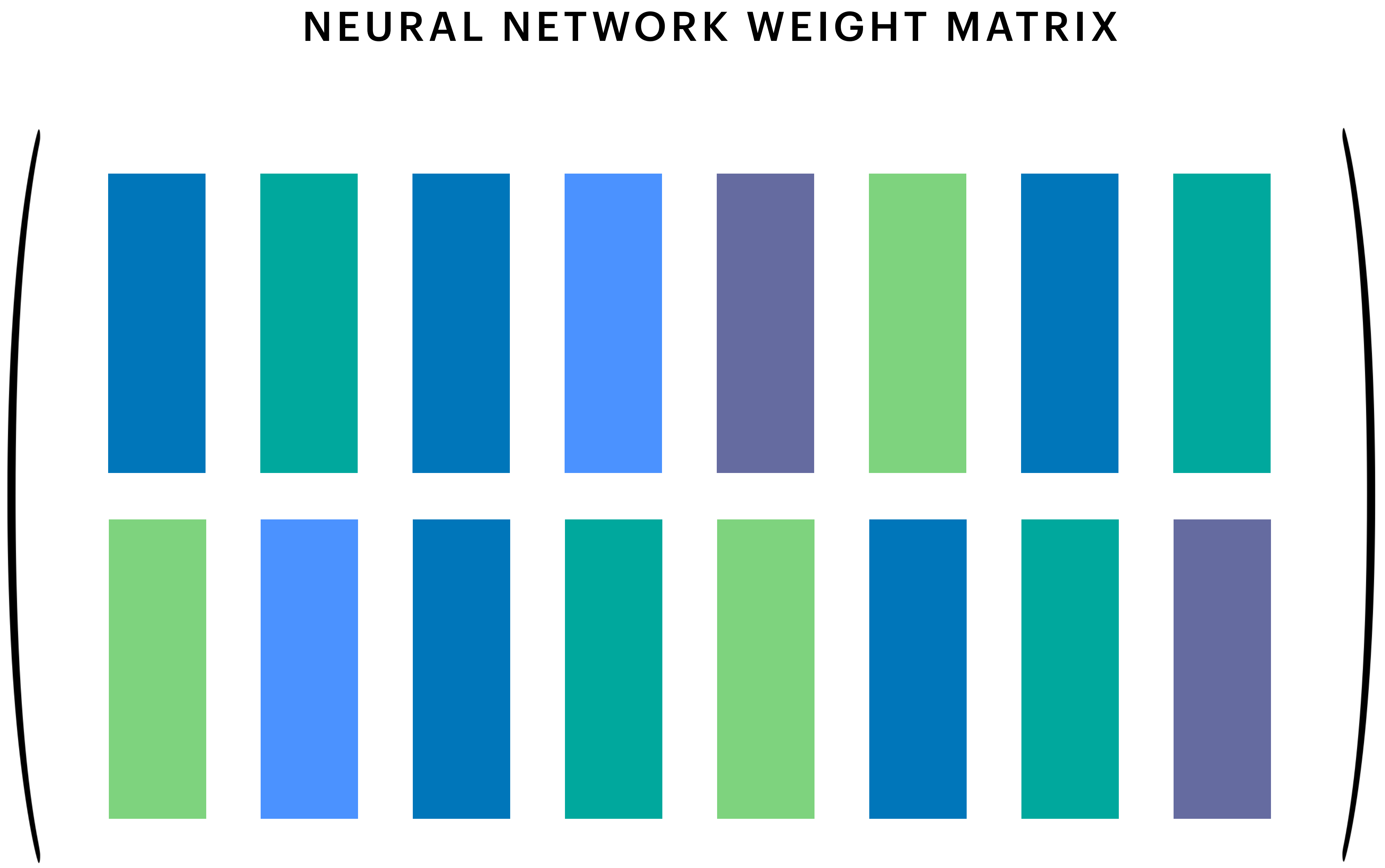
# Training a Smaller Model from Scratch

- Training Time ✓
- Inference Time ✓
- Model size ✓
- Performance ✗

# Techniques for Smaller Networks

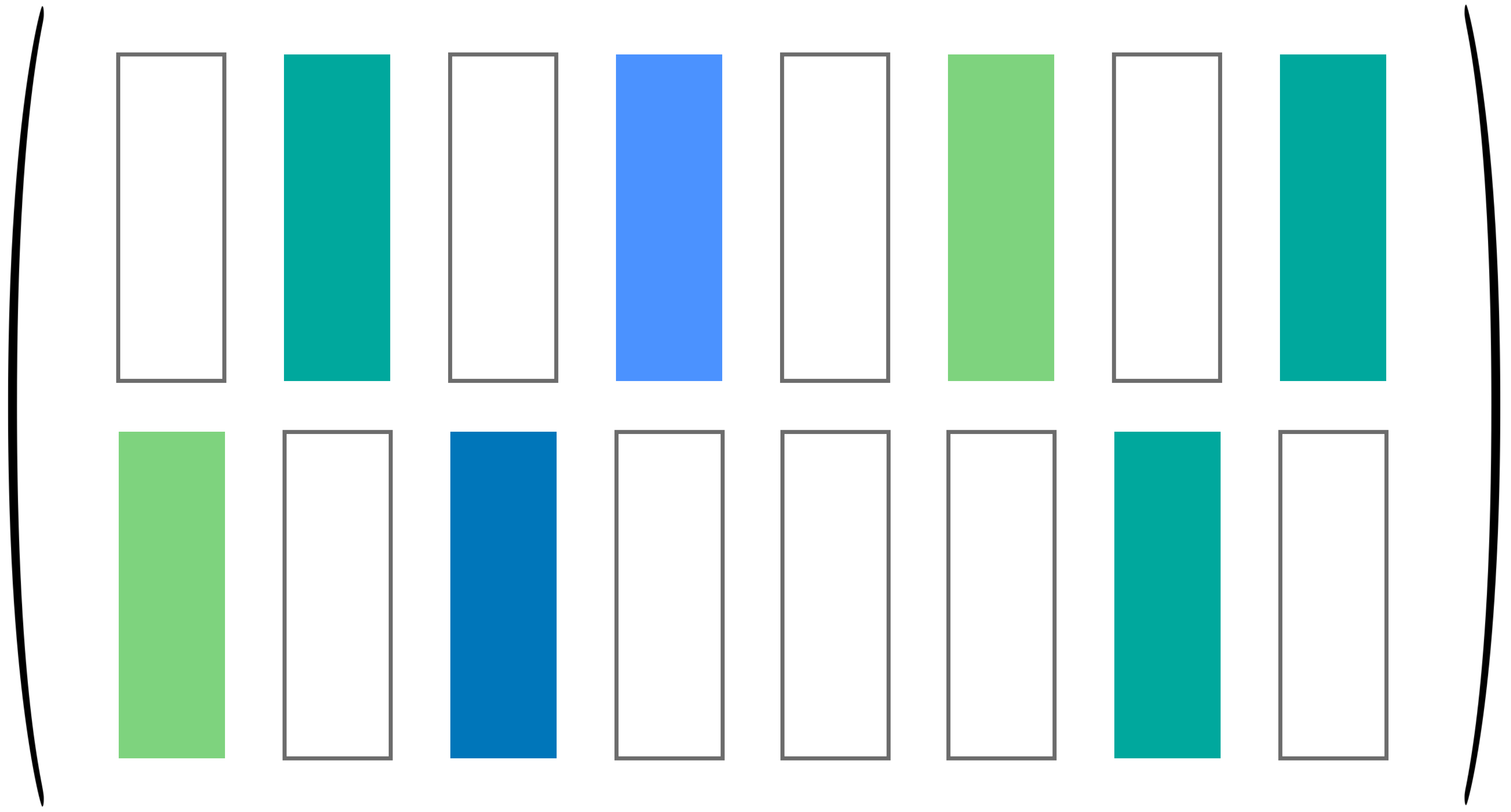
- Train Smaller Network from Scratch
- **Sparsity Inducing Training**
- Knowledge Distillation
- Pruning
- Weight Sharing
- Quantization
- More efficient architectures

# Sparsity Inducing Training



# Sparsity Inducing Training

NEURAL NETWORK WEIGHT MATRIX

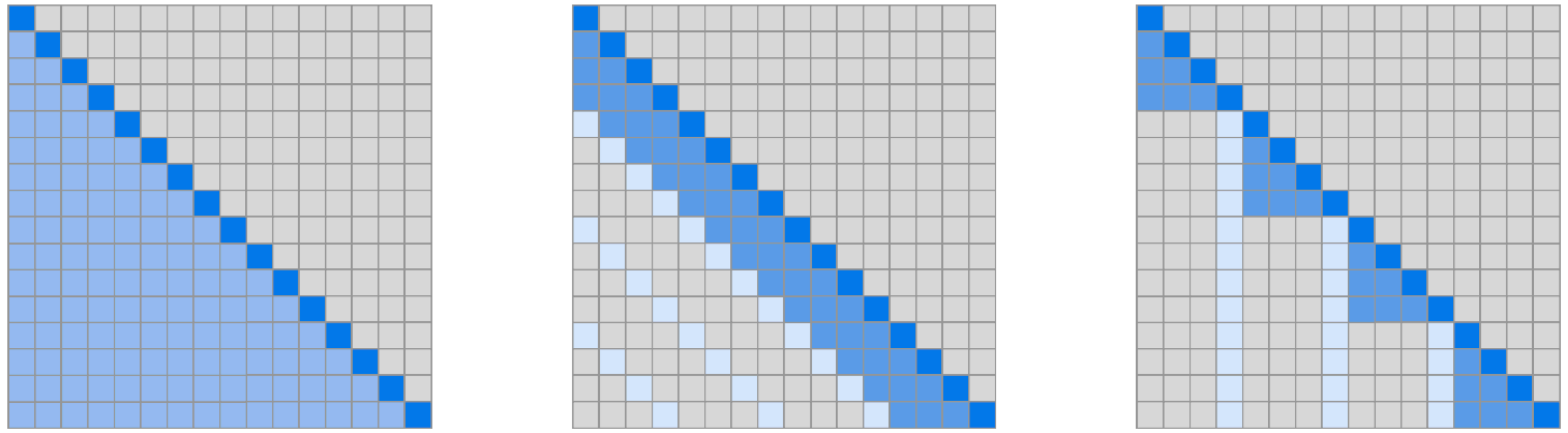


sparsify the weight matrix to include many zeroes

# Advantages

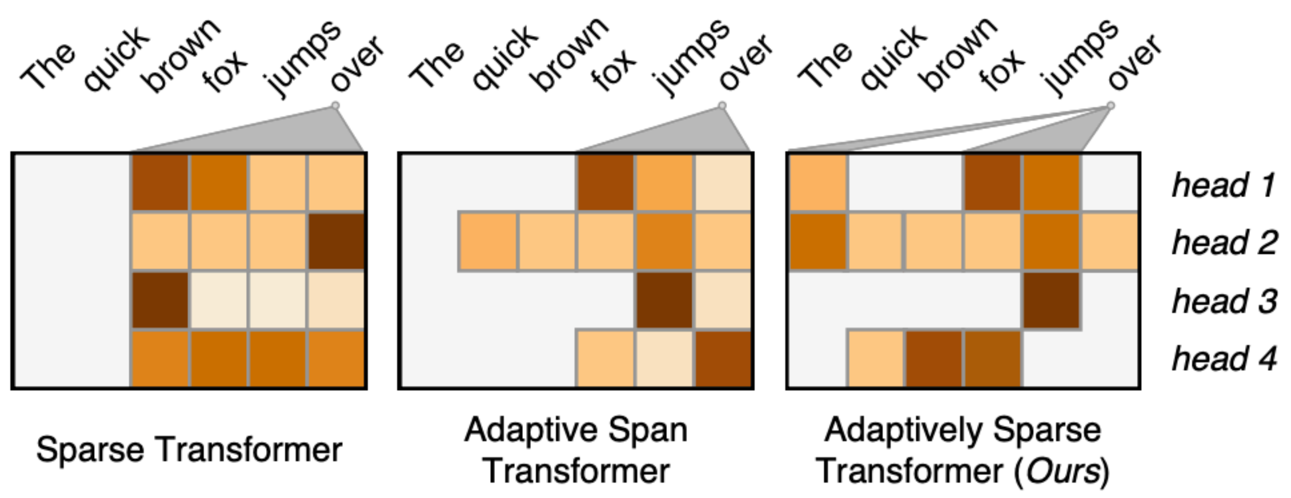
- Sparse matrix multiplication - takes advantage of the zeroes
- Specialized kernels - everywhere you see a zero, don't need to compute that row/column, so less multiplications
- Important for on-device

# Sparsity Inducing Training via Attention Matrices



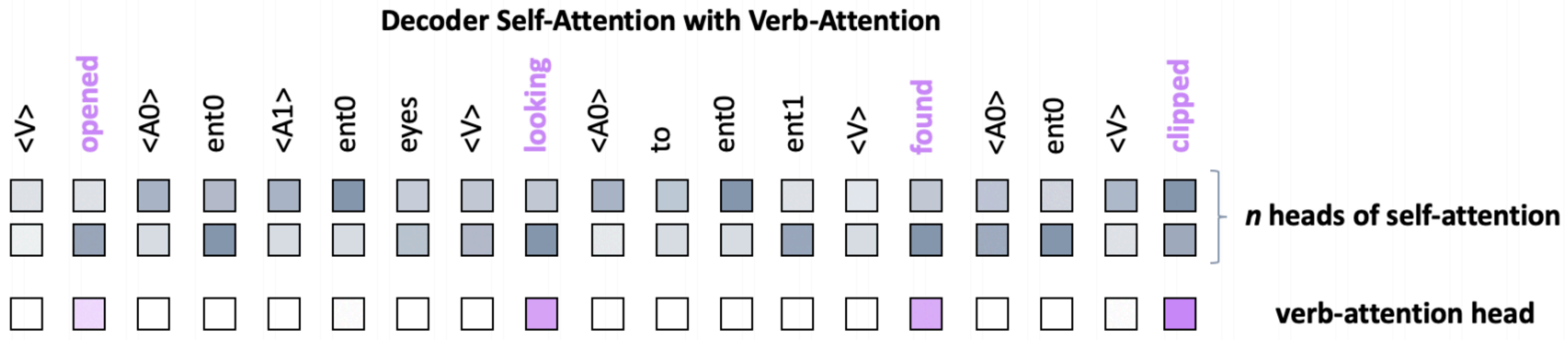
GENERATING LONG SEQUENCES WITH SPARSE TRANSFORMERS

CHILD ET AL




ADAPTIVELY SPARSE TRANSFORMERS

CORREIA ET AL

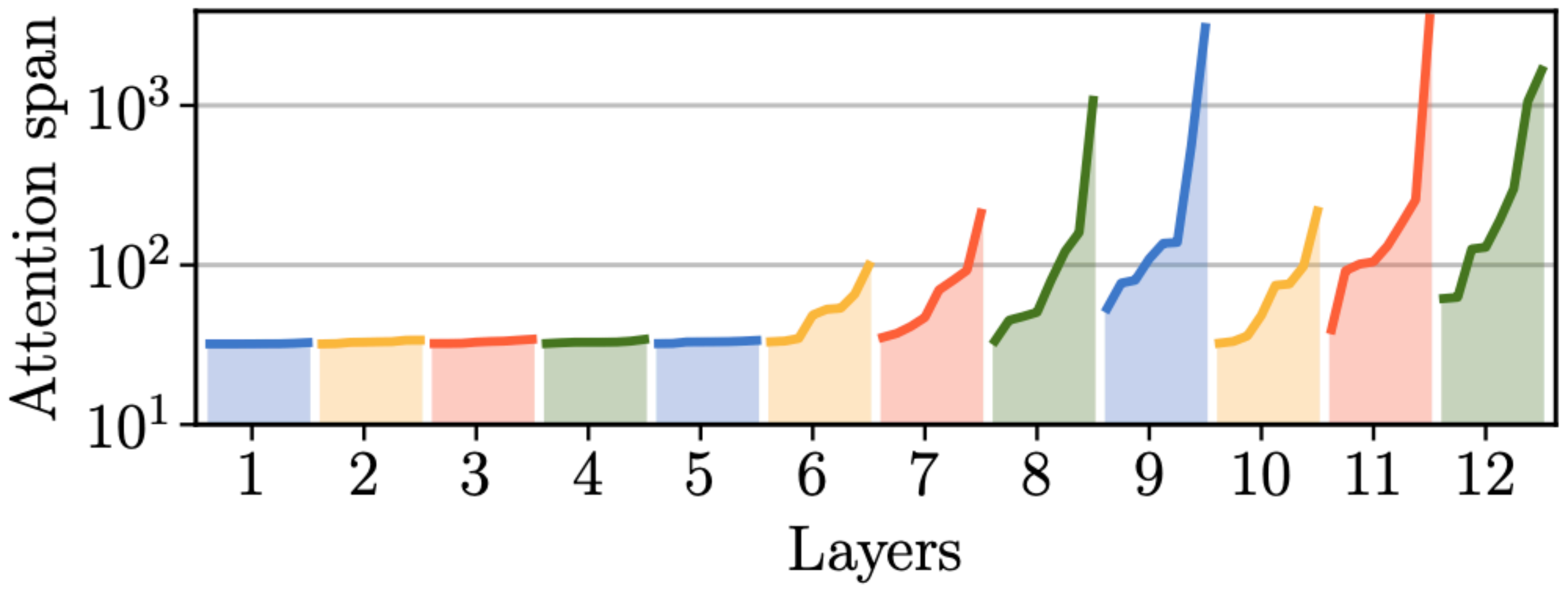



STRATEGIES FOR STRUCTURING STORY GENERATION

FAN ET AL

# Sparsity Inducing Training via Network Losses

penalize network for using parameters by increasing loss




ADAPTIVE ATTENTION SPAN IN TRANSFORMERS

SUKHBAATAR ET AL

# Sparsity Inducing Training

- Inference Time
- Energy
- Performance

at least, often no performance drop

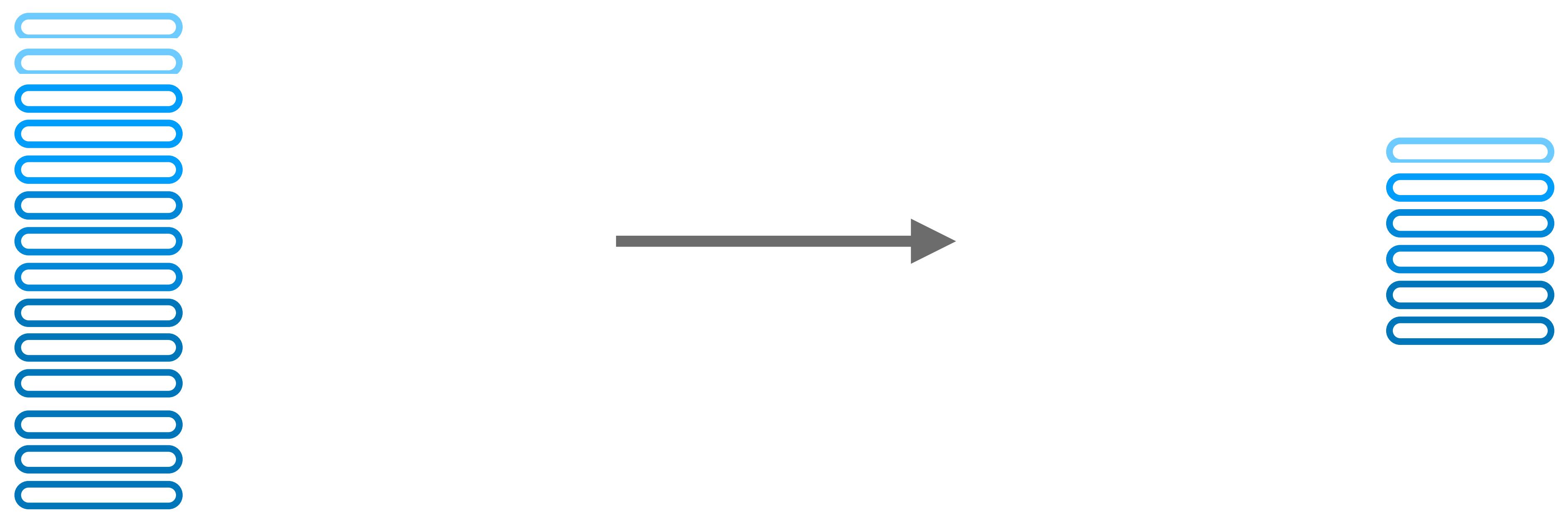
# Techniques for Smaller Networks

- Train Smaller Network from Scratch
- Sparsity Inducing Training
- **Knowledge Distillation**
- Pruning
- Weight Sharing
- Quantization
- More efficient architectures

# Knowledge Distillation

LARGE, ACCURATE NEURAL NETWORK

SMALLER NEURAL NETWORK

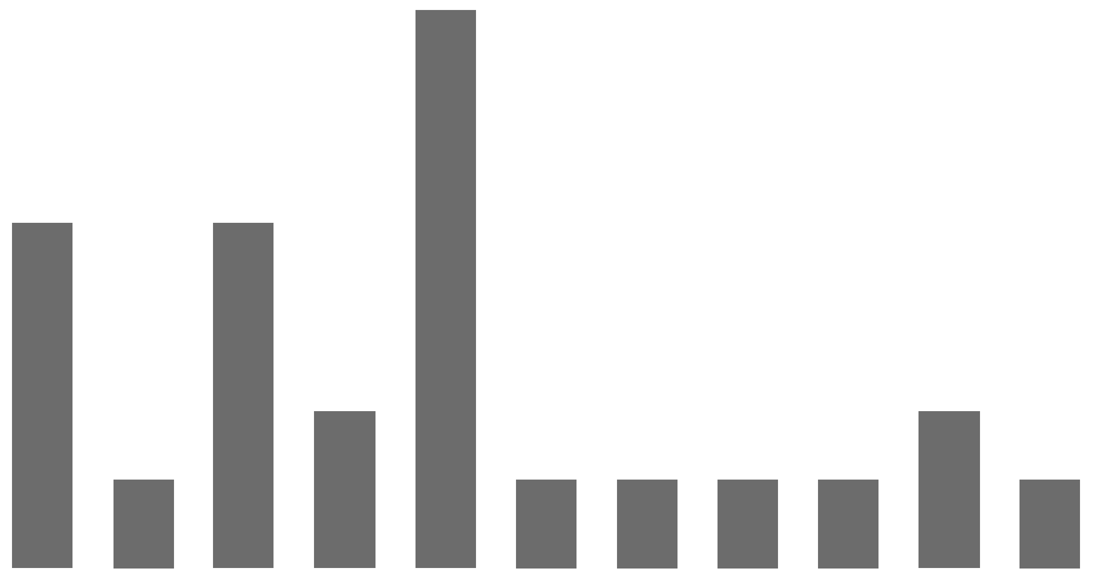


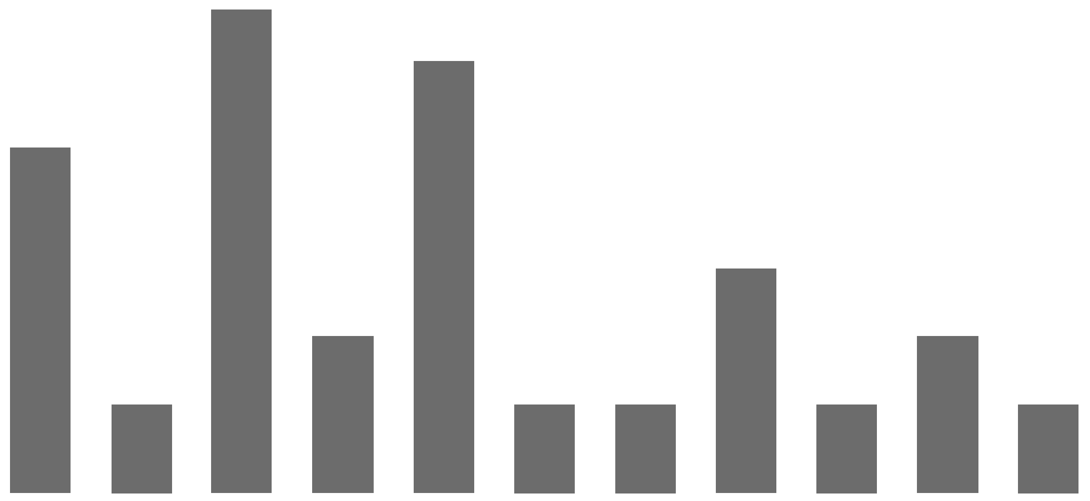
TEACHER MODEL

STUDENT MODEL

# Knowledge Distillation

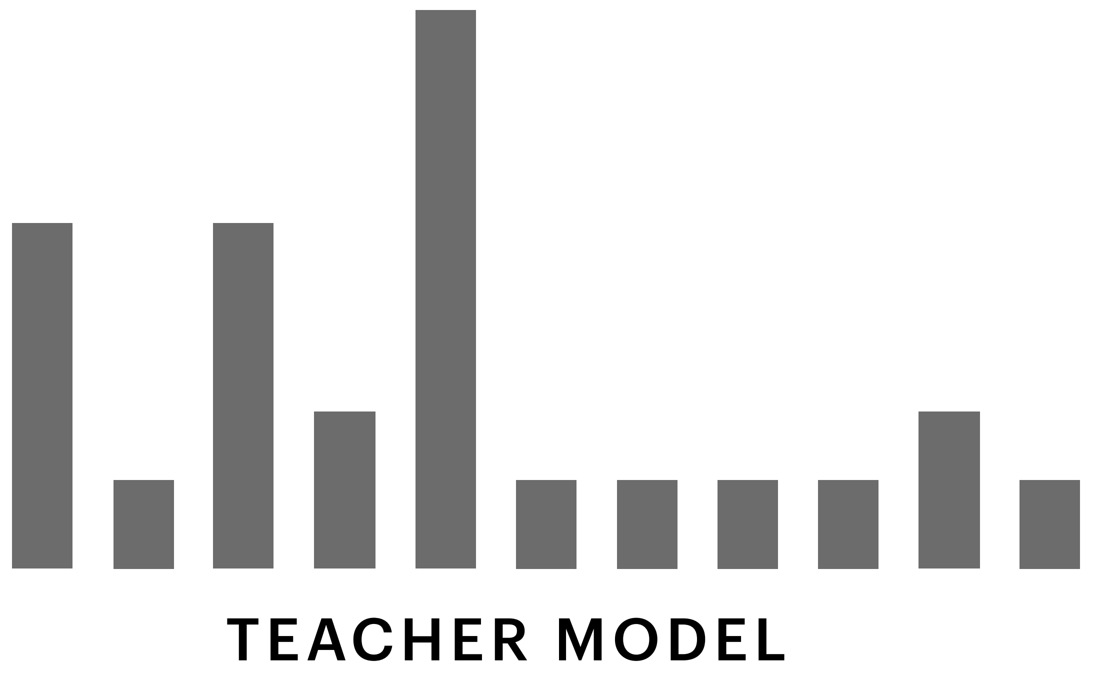
student model learns to mimic the output of the teacher model

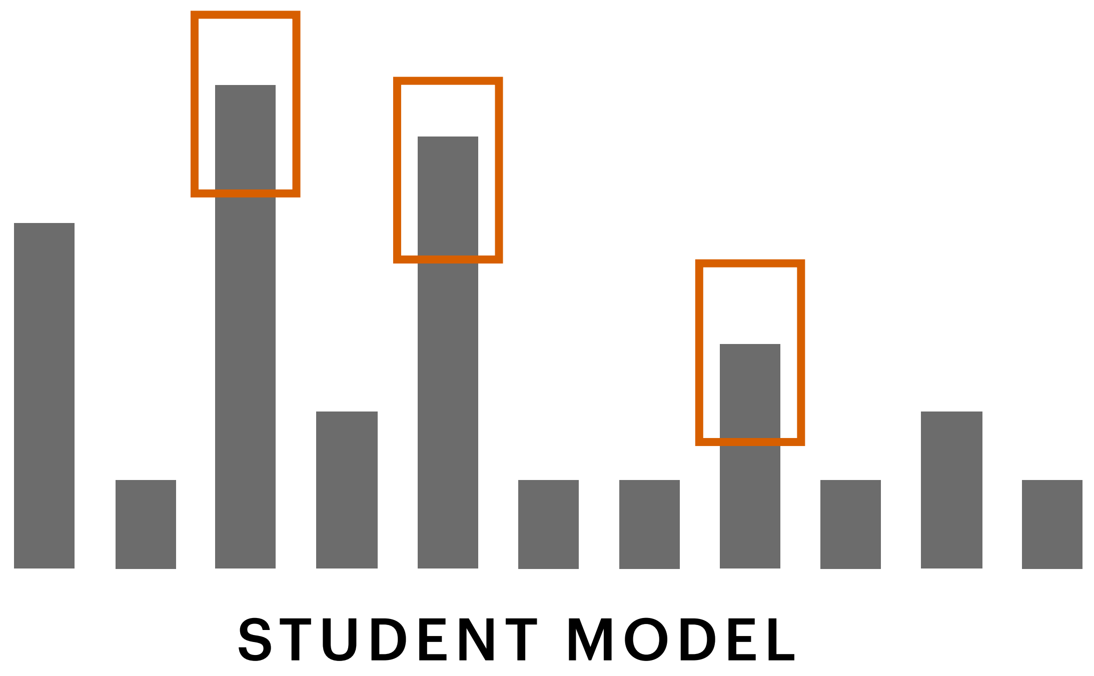


# Knowledge Distillation

student model learns to mimic the output of the teacher model




CALCULATE DIFFERENCE BETWEEN THE TWO PREDICTIONS

# Advantages

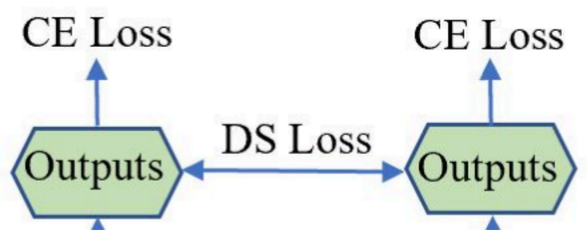
- Flexibility over size - teacher and student can both be any size

# Advantages

- Flexibility over size - teacher and student can both be any size
- Not limited to the training data - any data can be used to distill
    - data augmentation is **very powerful**

# Advantages

- Flexibility over size - teacher and student can both be any size
- Not limited to the training data - any data can be used to distill
- Can also learn from intermediate layers

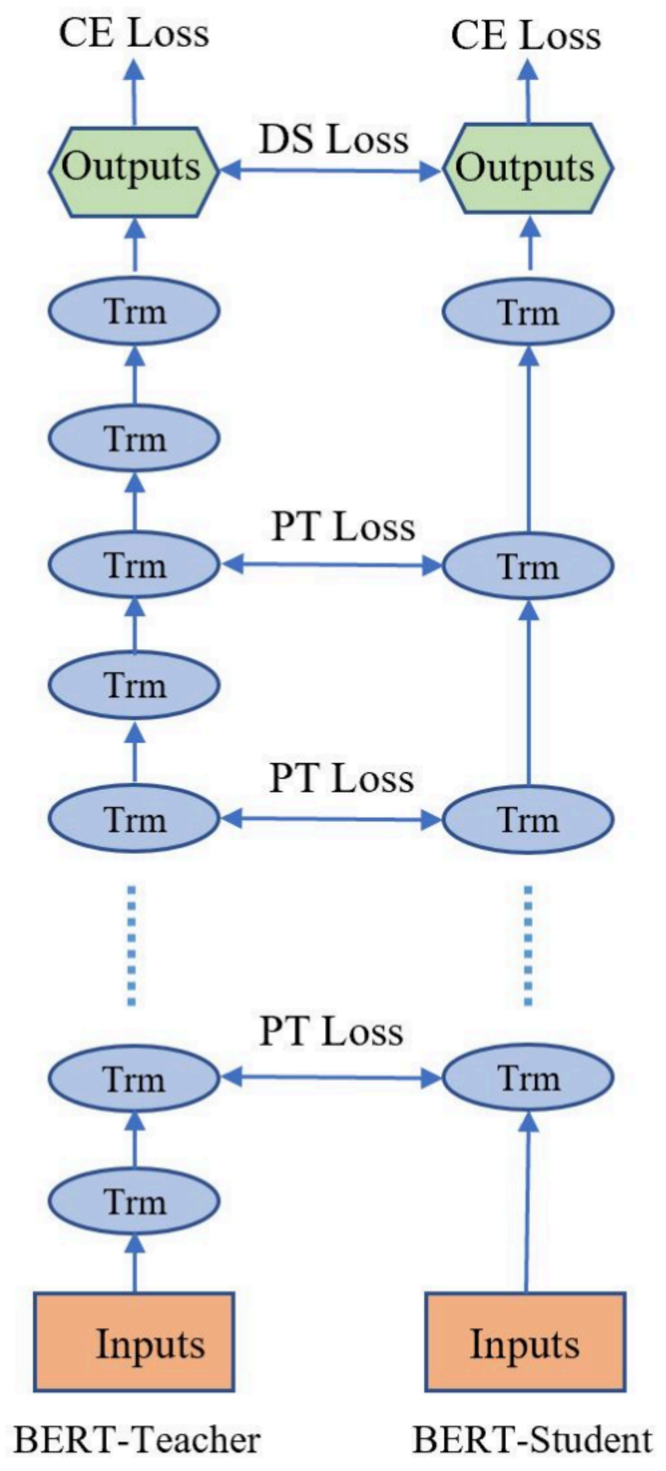



PATIENT KNOWLEDGE DISTILLATION FOR BERT MODEL COMPRESSION

SUN ET AL

# Advantages

- Flexibility over size - teacher and student can both be any size
- Not limited to the training data - any data can be used to distill
- Can also learn from intermediate layers




PATIENT KNOWLEDGE DISTILLATION FOR BERT MODEL COMPRESSION

SUN ET AL

# Knowledge Distillation

- Training Time
- Inference Time
- Performance

training time: need pre-trained teacher, but often state of the art large models can be downloaded

# Knowledge Distillation

people love knowledge distillation!

TINYBERT: DISTILLING BERT FOR NATURAL LANGUAGE UNDERSTANDING
JIAO ET AL

DISTILBERT, A DISTILLED VERSION OF BERT: SMALLER, FASTER, CHEAPER AND LIGHTER
SANH ET AL

WELL-READ STUDENTS LEARN BETTER: ON THE IMPORTANCE OF PRE-TRAINING COMPACT MODELS
TURC ET AL

MOBILEBERT: TASK-AGNOSTIC COMPRESSION OF BERT BY PROGRESSIVE KNOWLEDGE TRANSFER
SUN ET AL

AND MORE!

# Techniques for Smaller Networks

- Train Smaller Network from Scratch
- Sparsity Inducing Training
- Knowledge Distillation
- **Pruning** remove layers from a trained model
- Weight Sharing
- Quantization
- More efficient architectures

# Techniques for Smaller Networks

- Train Smaller Network from Scratch
- Sparsity Inducing Training

if you want a smaller model size, need to re-train

- Knowledge Distillation
- **Pruning (with LayerDrop)**
- Weight Sharing
- Quantization
- More efficient architectures

# How can we make models smaller without re-training?

**Goal**: Train One Network, Prune to Any Depth at Inference Time

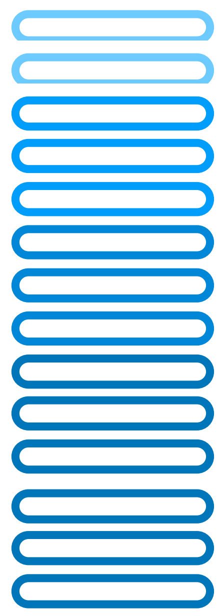
# How can we make models smaller without re-training?

**Goal**: Train One Network, Prune to Any Depth at Inference Time

↓

Drop Any Layer and Model Remains the Same

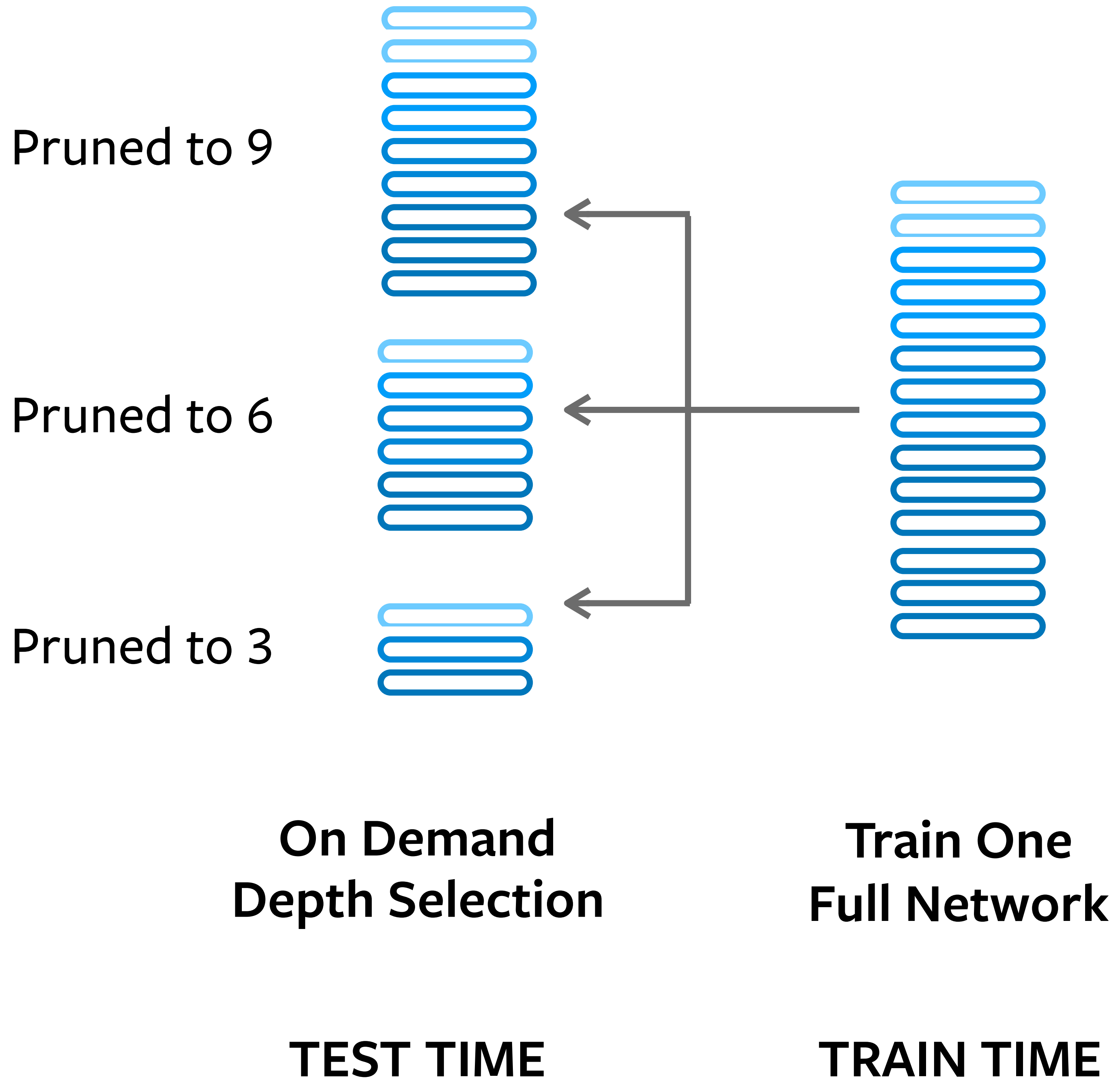
# How can we make models smaller without re-training?



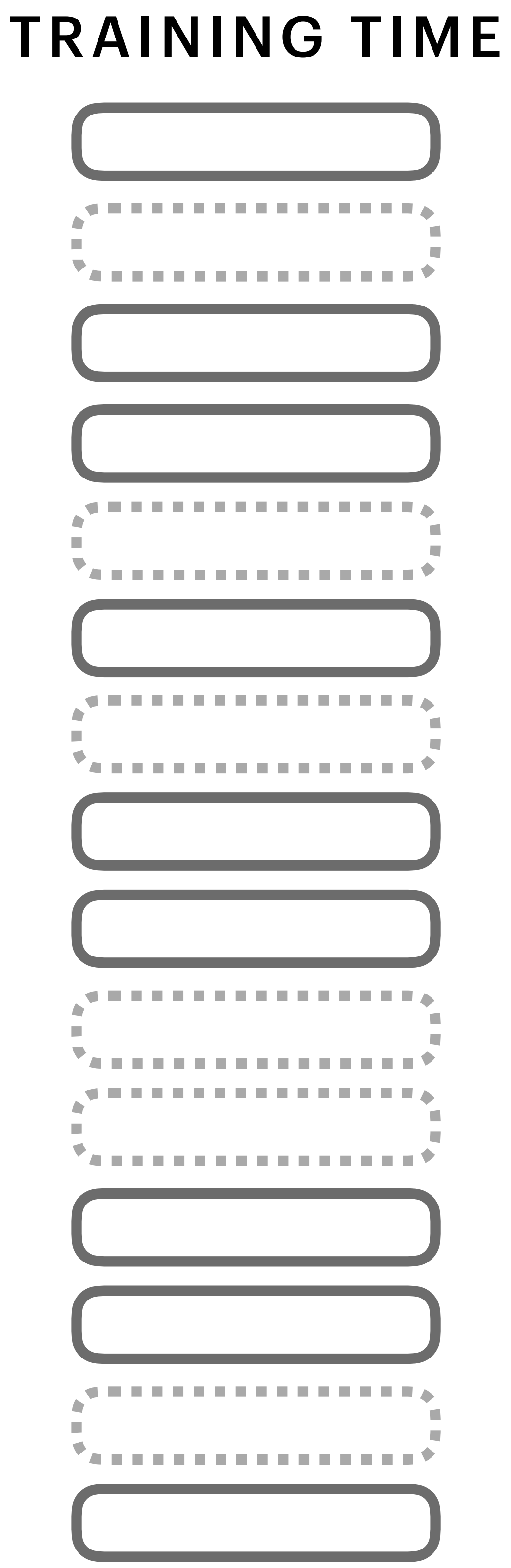
**Train One**
**Full Network**

**TRAIN TIME**

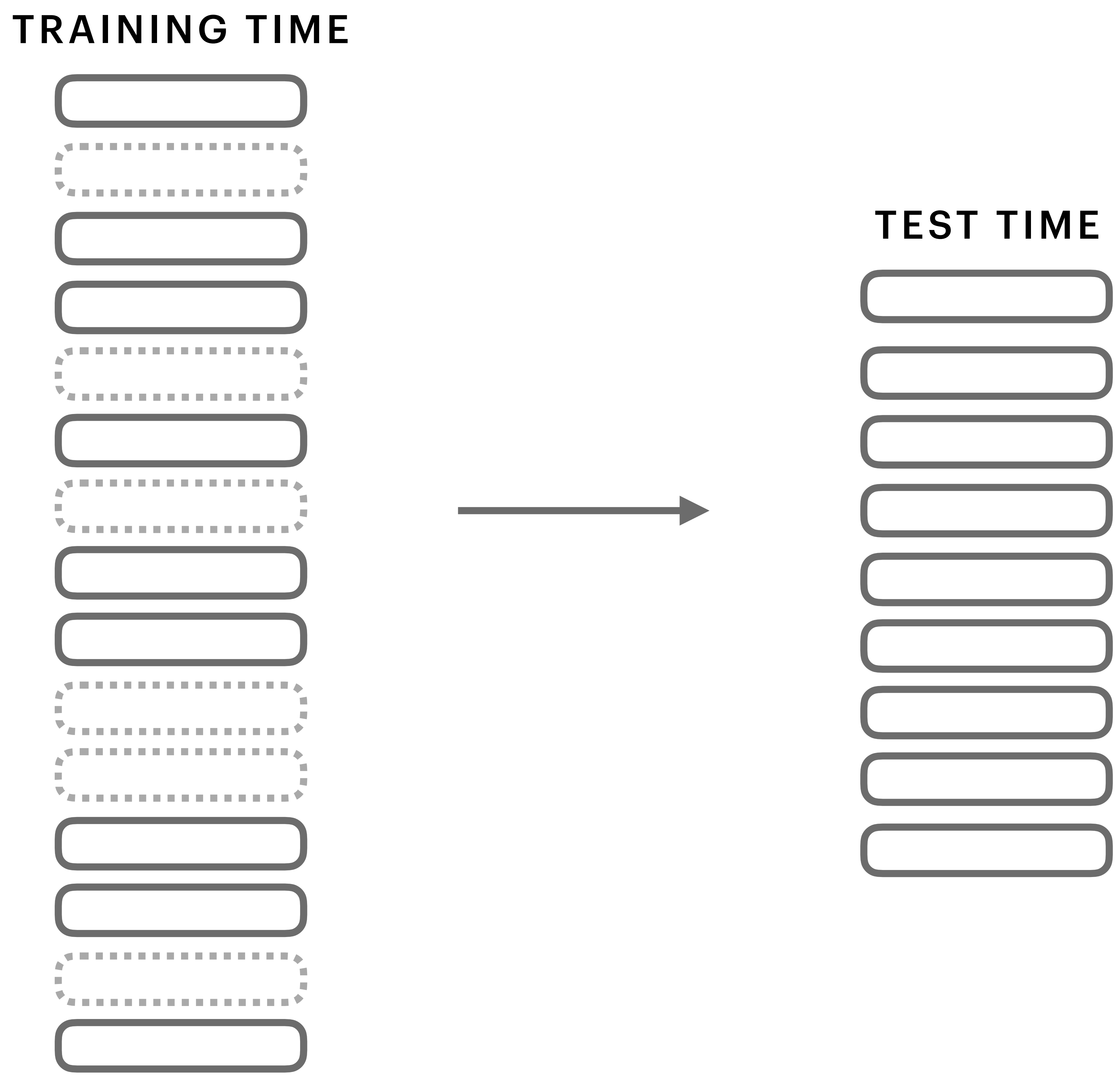
# How can we make models smaller without re-training?



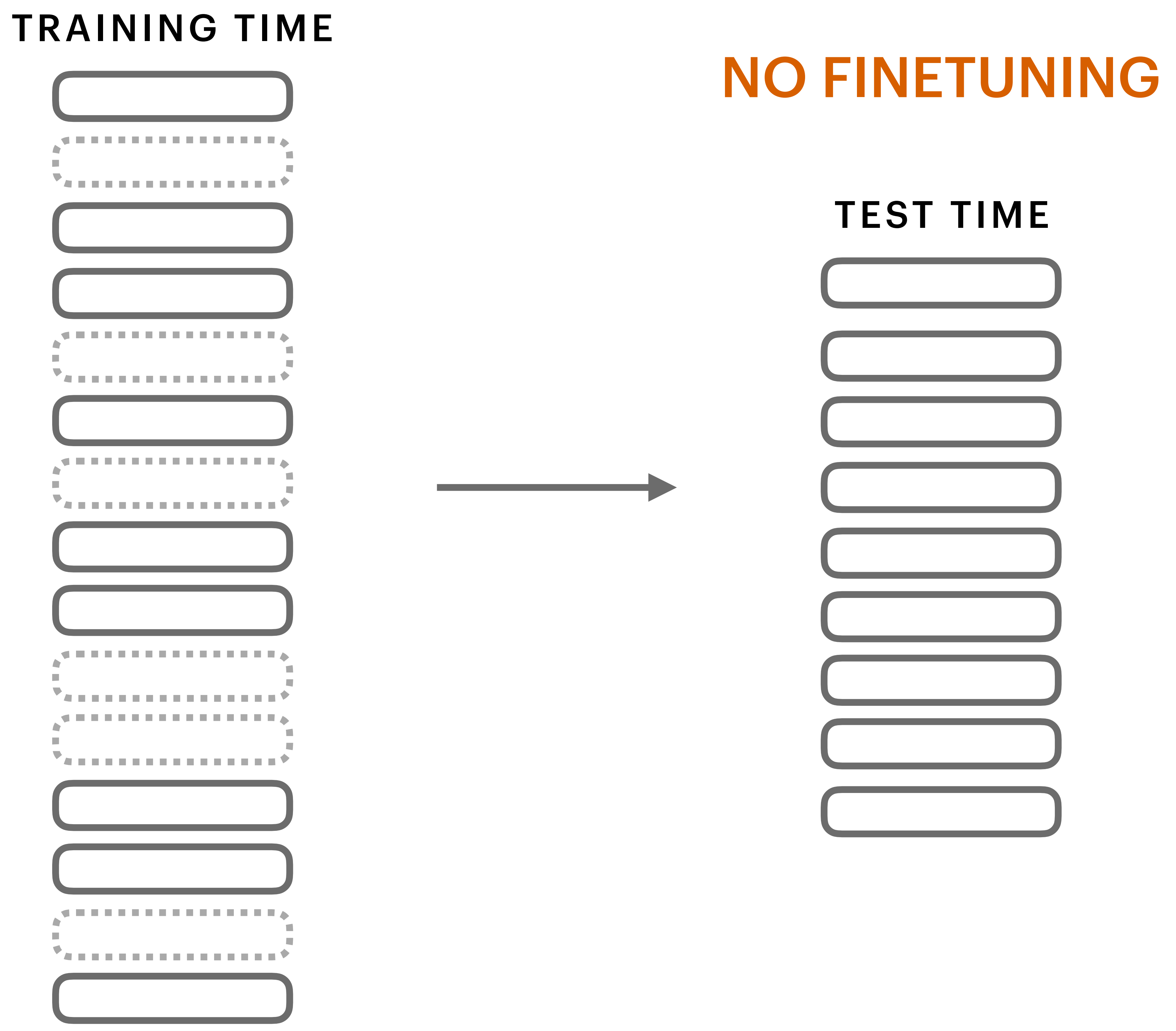
# Our Proposal: LayerDrop



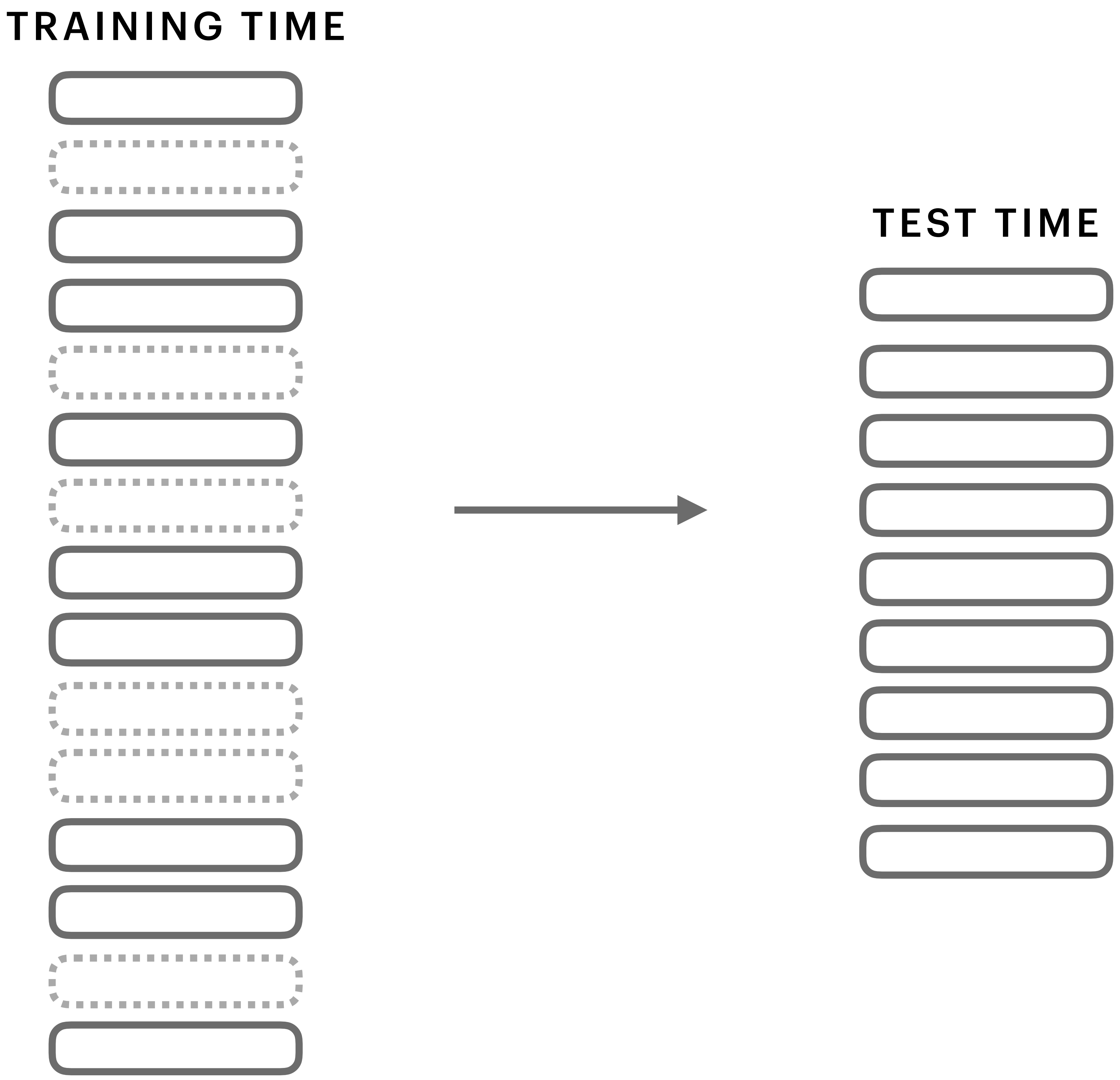
# Our Proposal: LayerDrop



# Our Proposal: LayerDrop

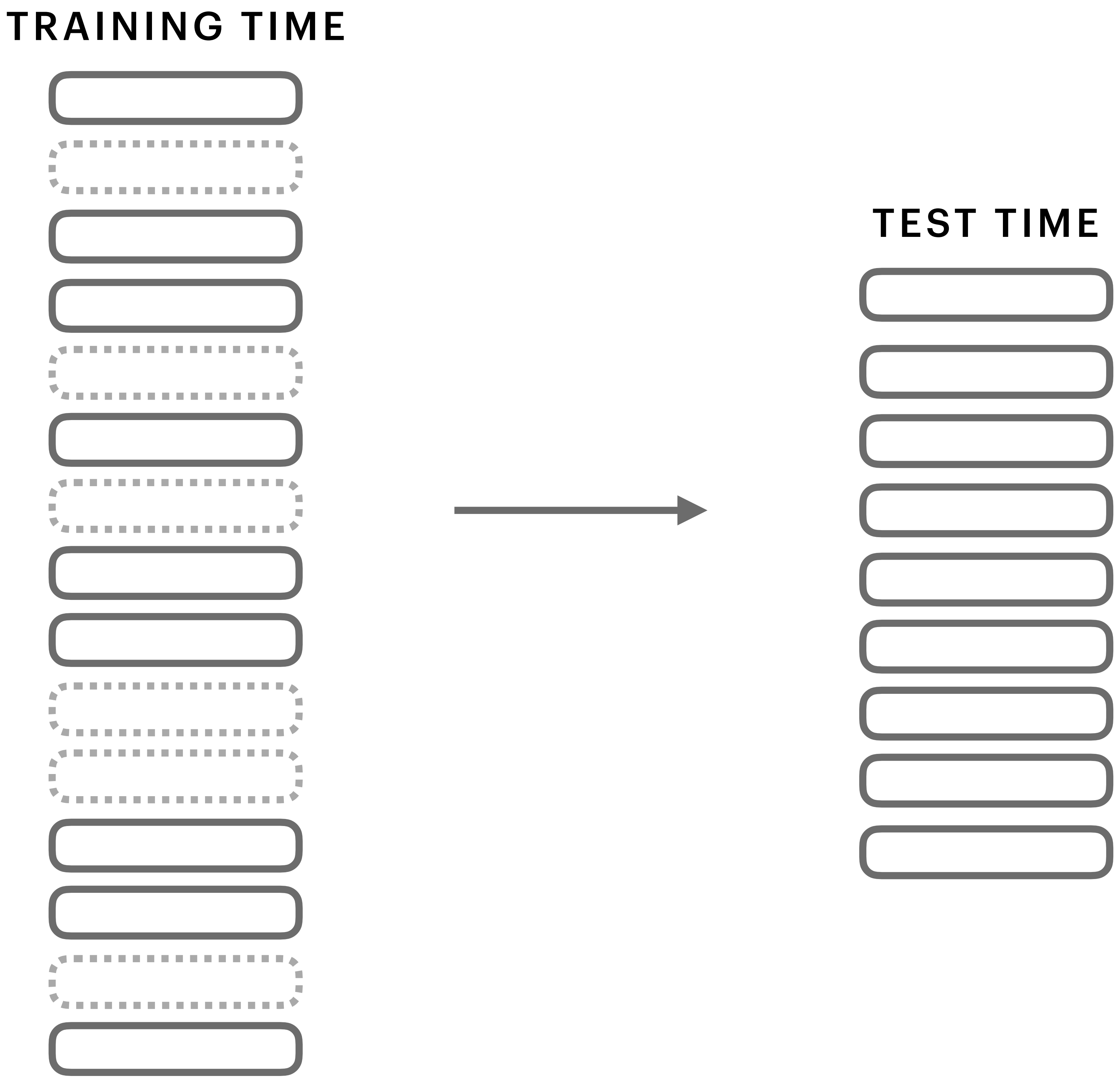


# Structured Dropout can be More General




- Layers (our focus)

# Structured Dropout can be More General

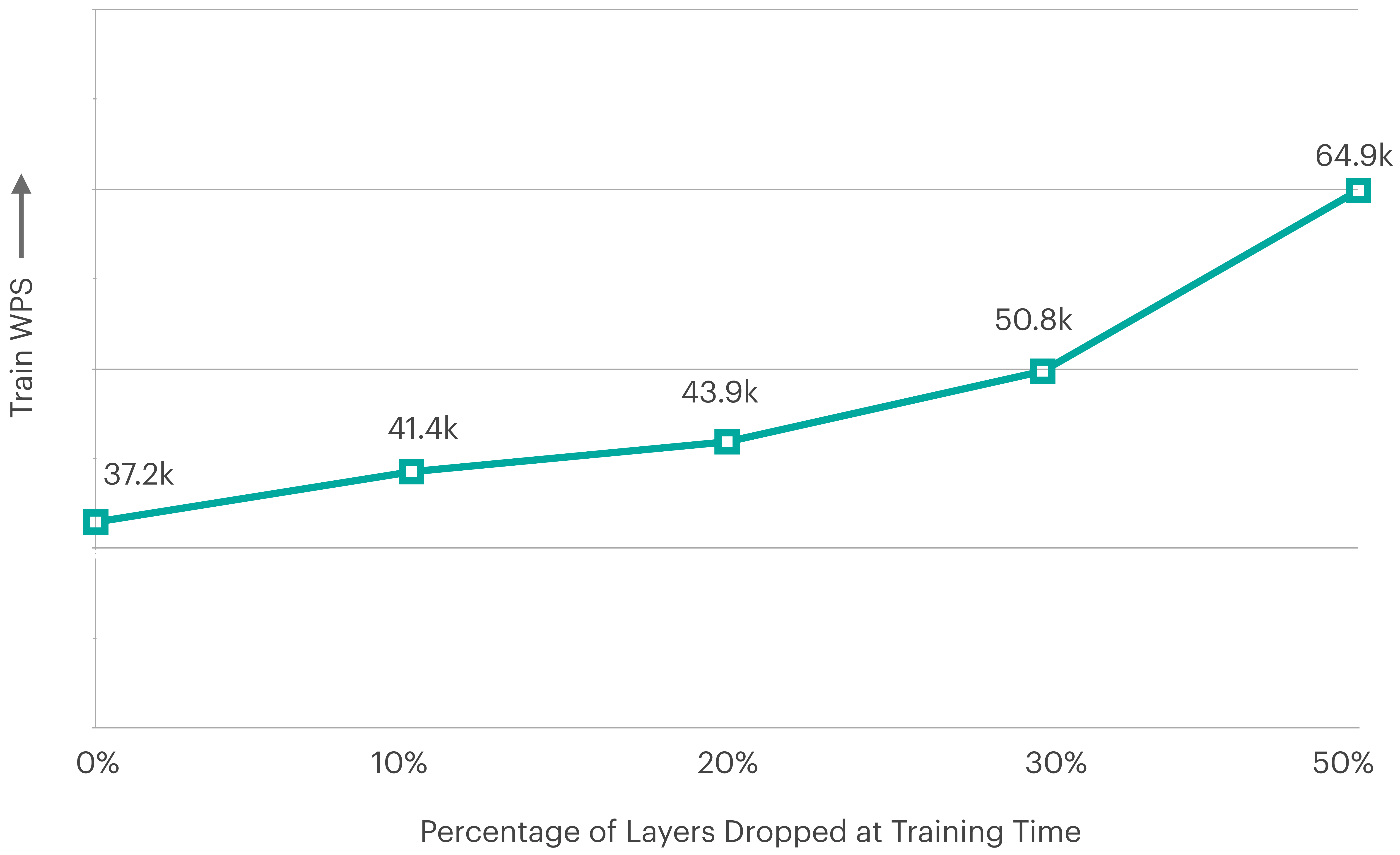



- Layers
- Attention Heads
- FFN, Attention Sub-Layers
- Portions of Matrices
- etc

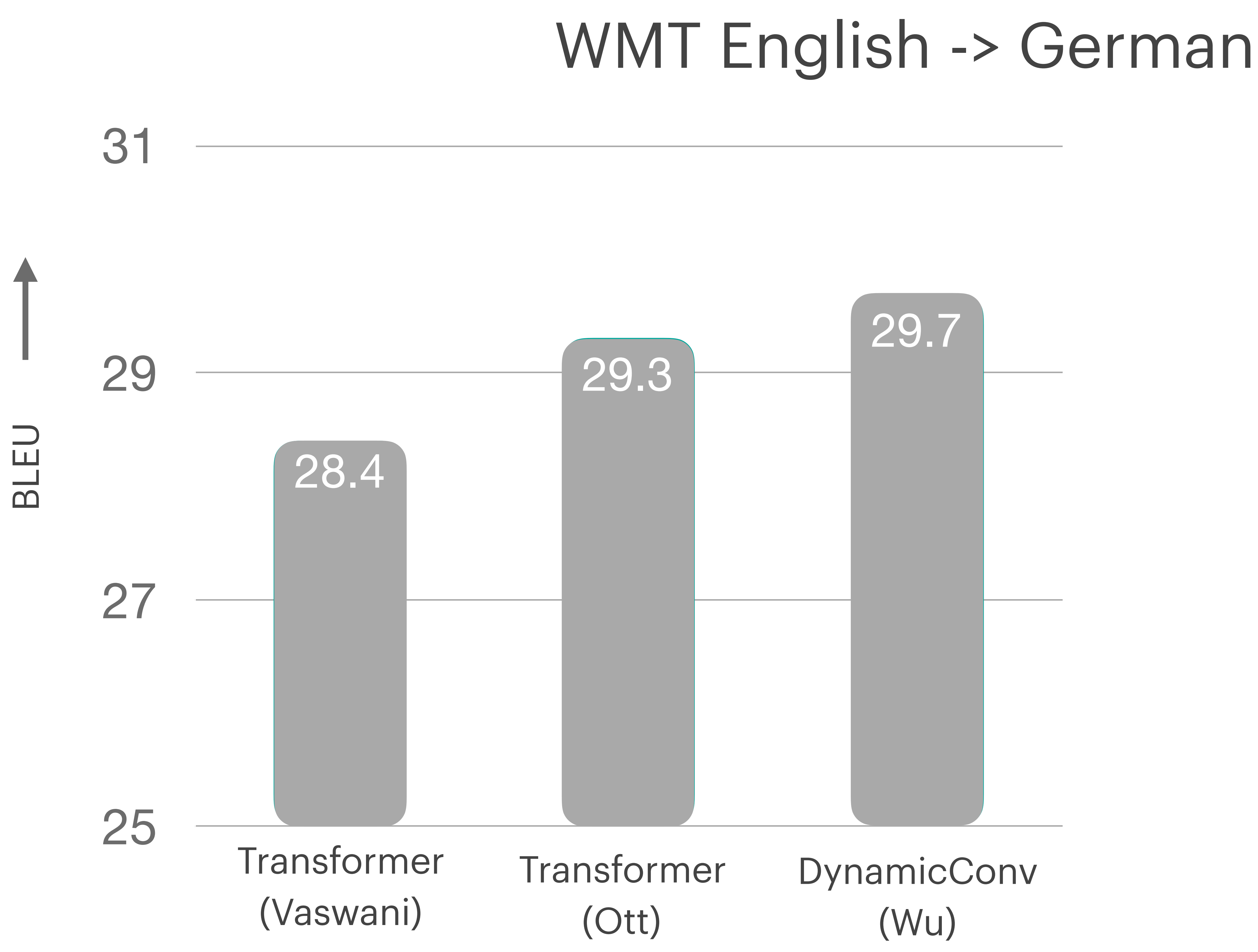
# Advantages

- Training Speed
- Regularization
- Reduction

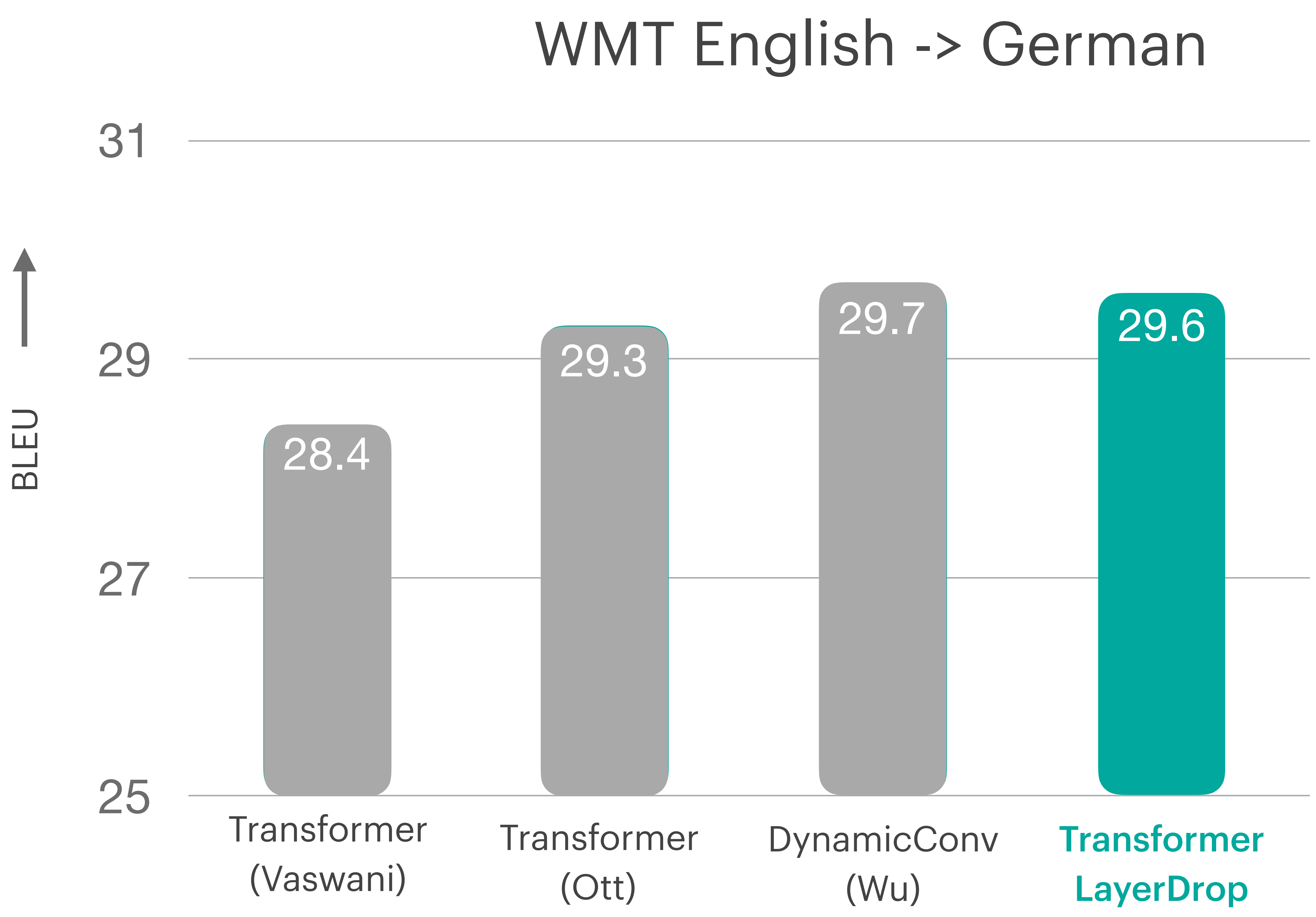
# (1) LayerDrop Increases Training Speed



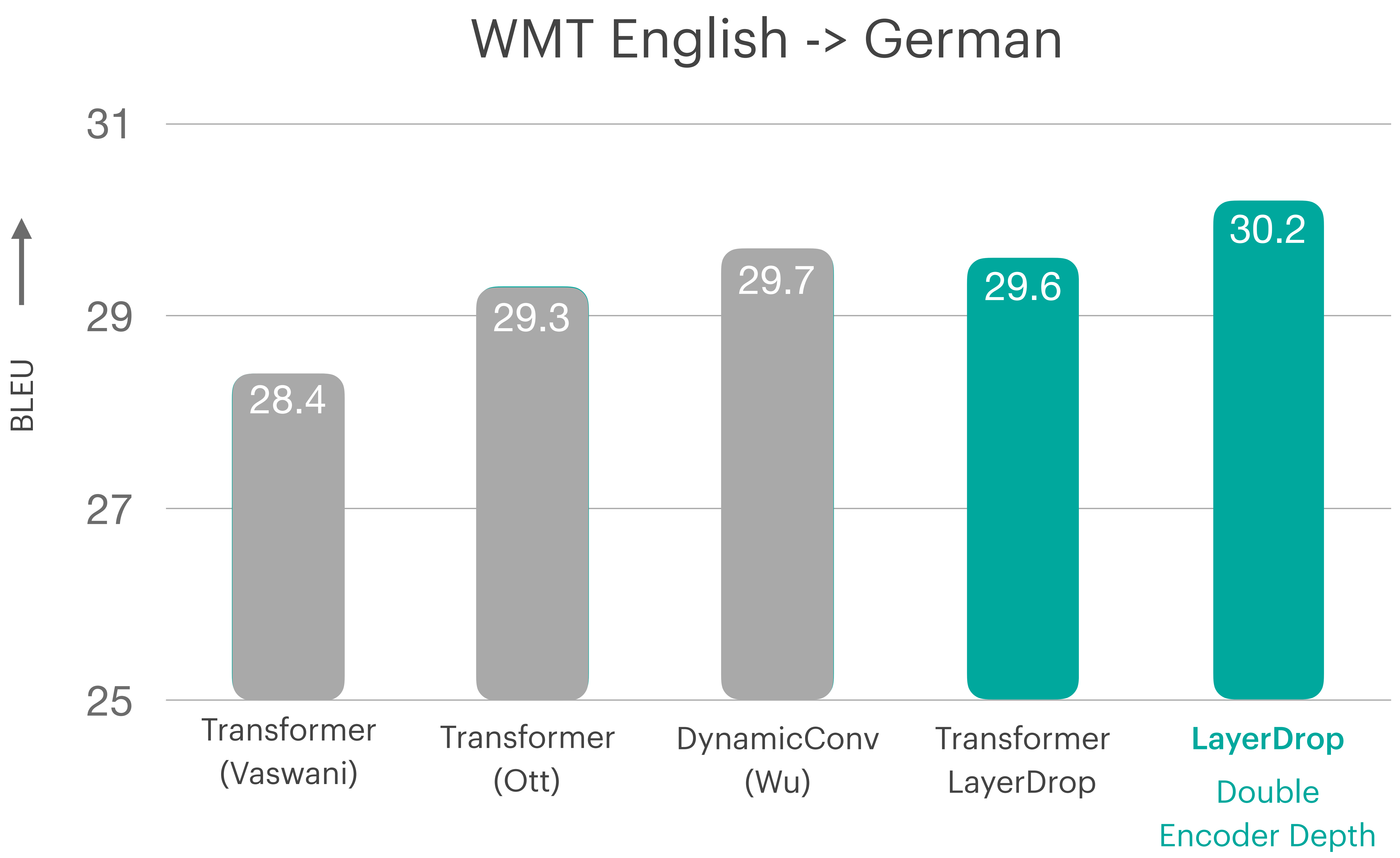
# (2) LayerDrop is an effective regularizer - Neural Machine Translation



# (2) LayerDrop is an effective regularizer - Neural Machine Translation



# (2) LayerDrop is an effective regularizer - Neural Machine Translation



# (3) Our Main Focus: LayerDrop for Pruning

- Train once, prune to any desired depth

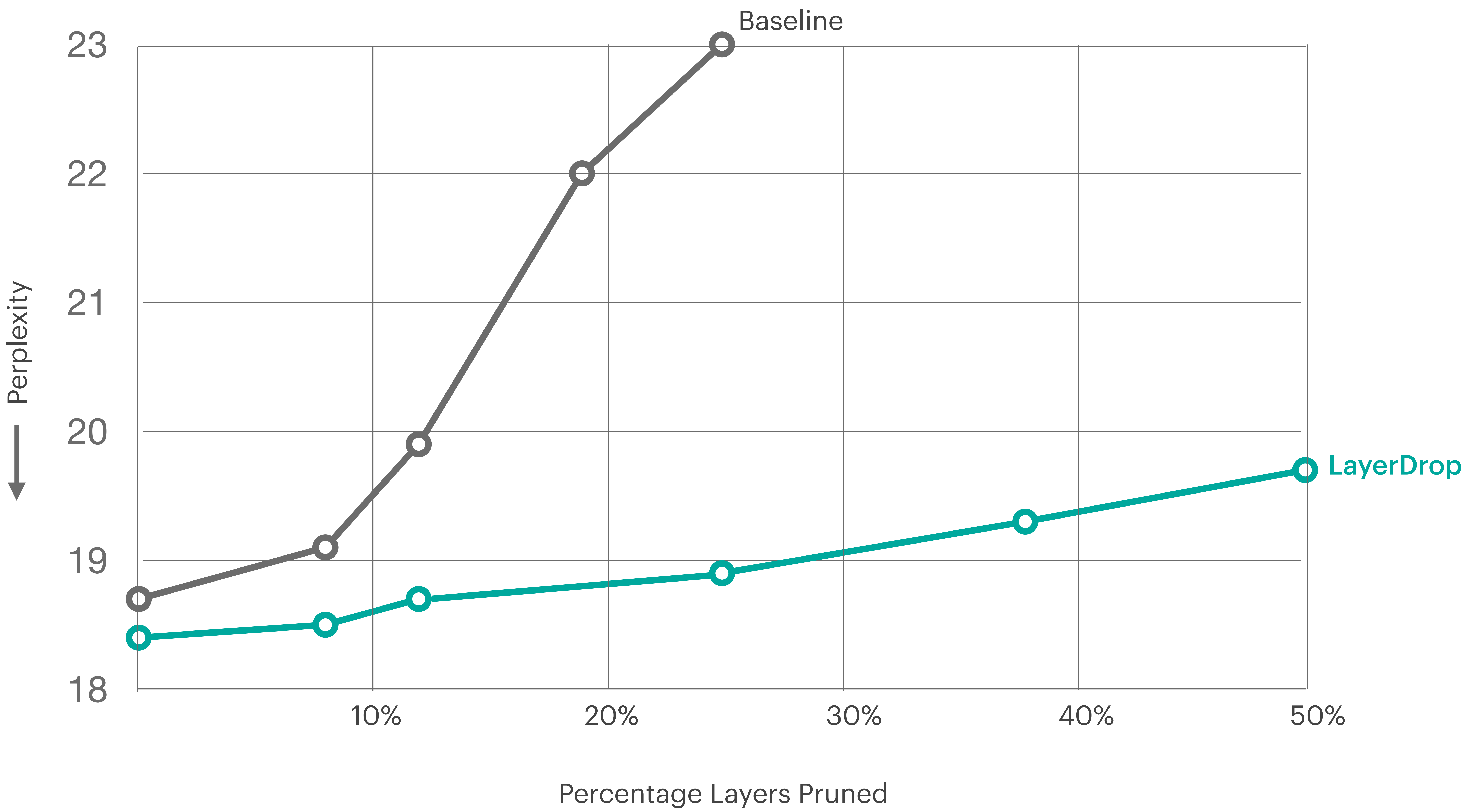
# (3) Our Main Focus: LayerDrop for Pruning

- Train once, prune to any desired depth
- Robust to parameter setting
  - use the same value for all experiments

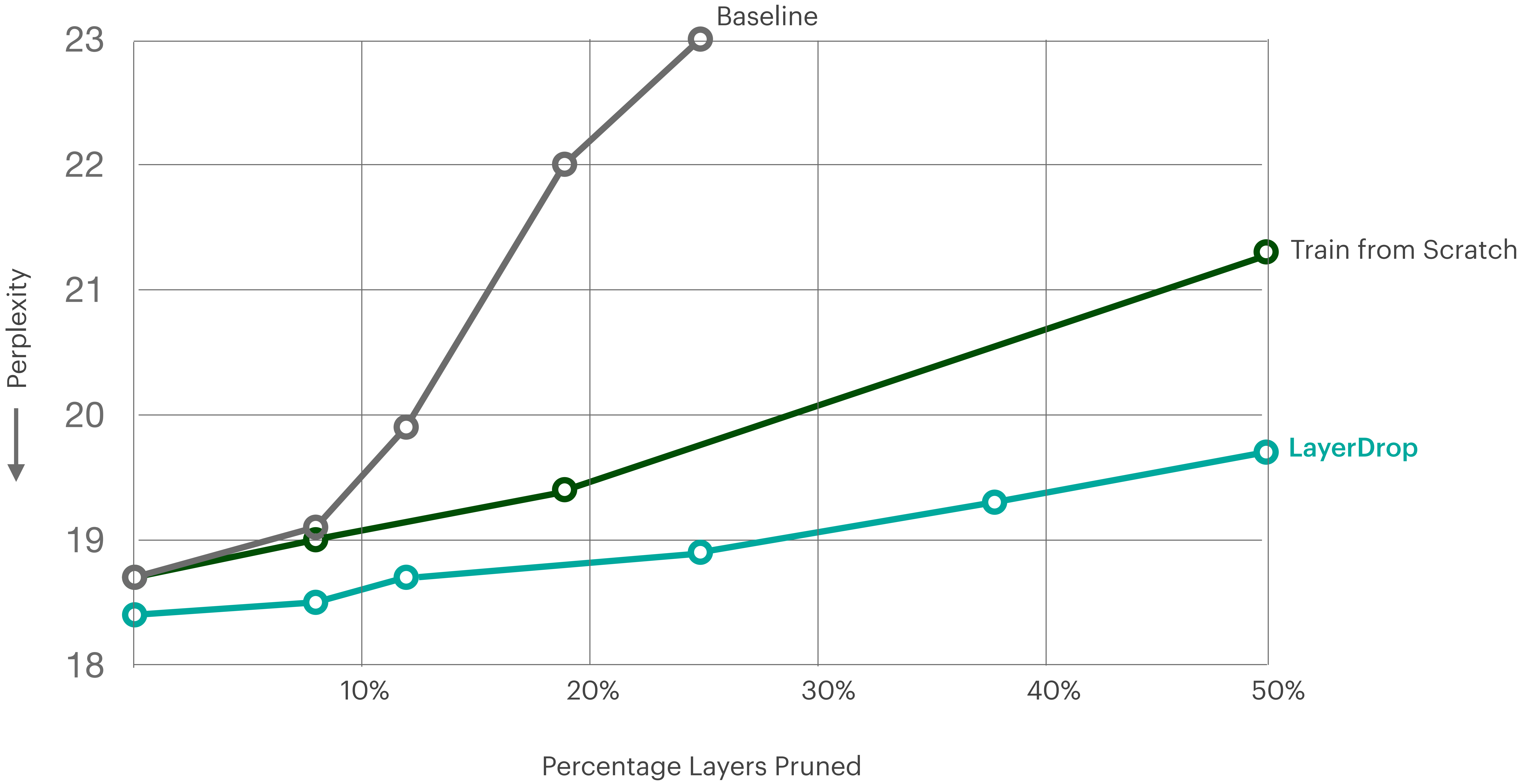
# (3) Our Main Focus: LayerDrop for Pruning

- Train once, prune to any desired depth
- Robust to parameter setting
- Specific Pruning Strategy is not Important

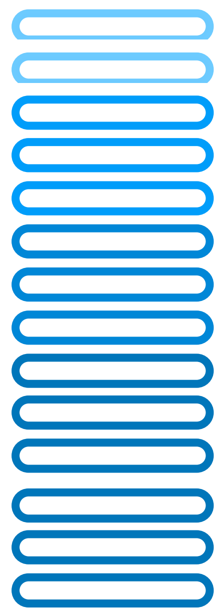
# (3) LayerDrop for Pruning - Language Modeling



# (3) LayerDrop for Pruning - Language Modeling

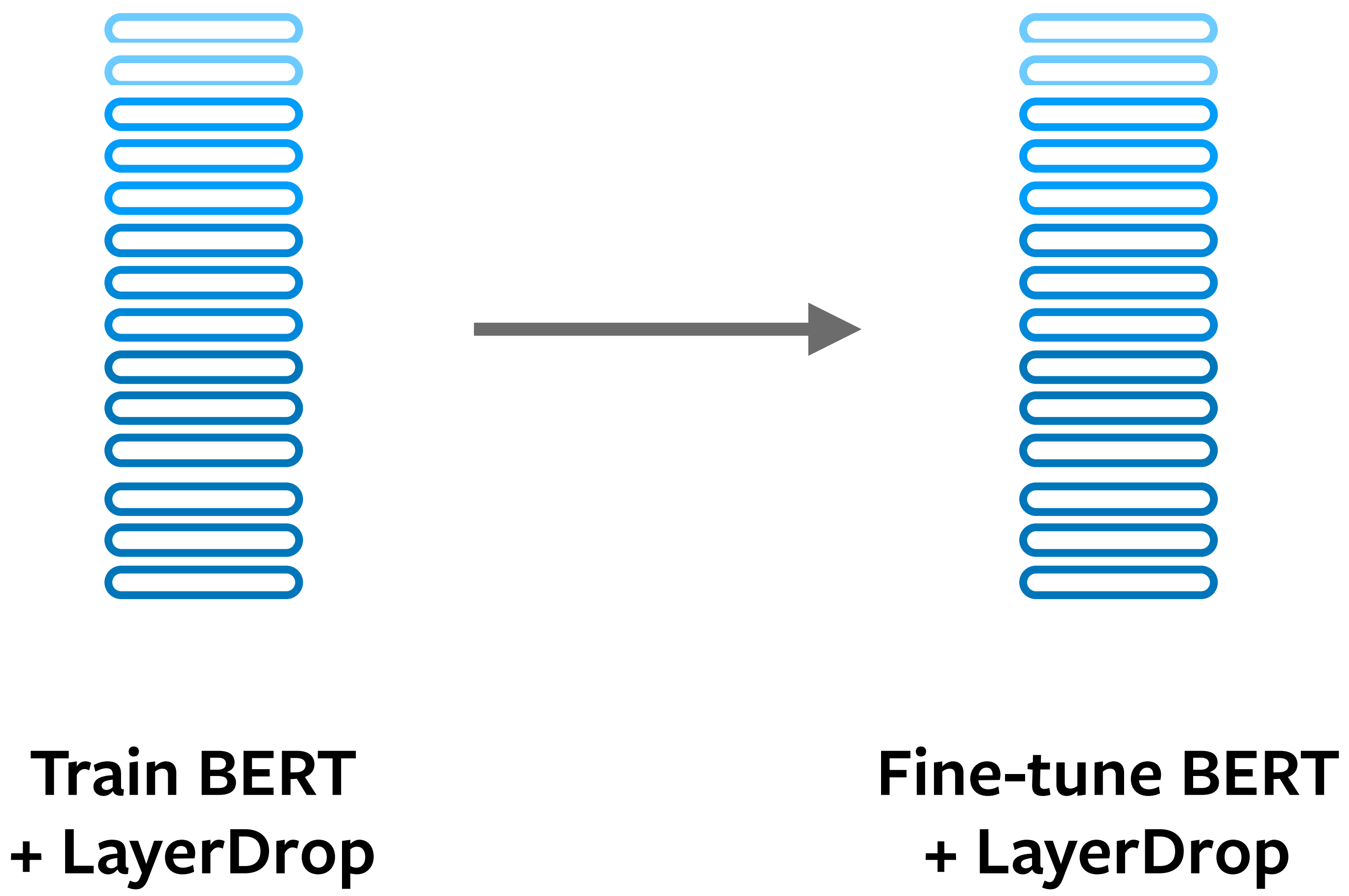


# (3) LayerDrop for Pruning - BERT Pre-Training

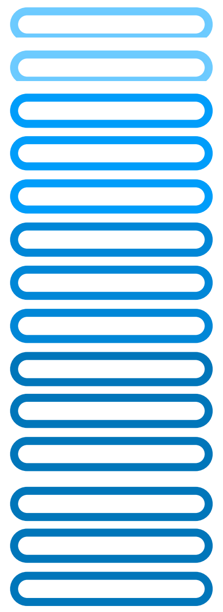


**Train BERT**
**+ LayerDrop**

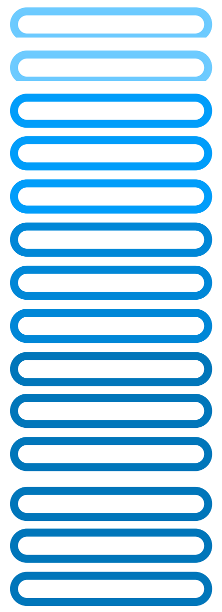
# (3) LayerDrop for Pruning - BERT Pre-Training



# (3) LayerDrop for Pruning - BERT Pre-Training



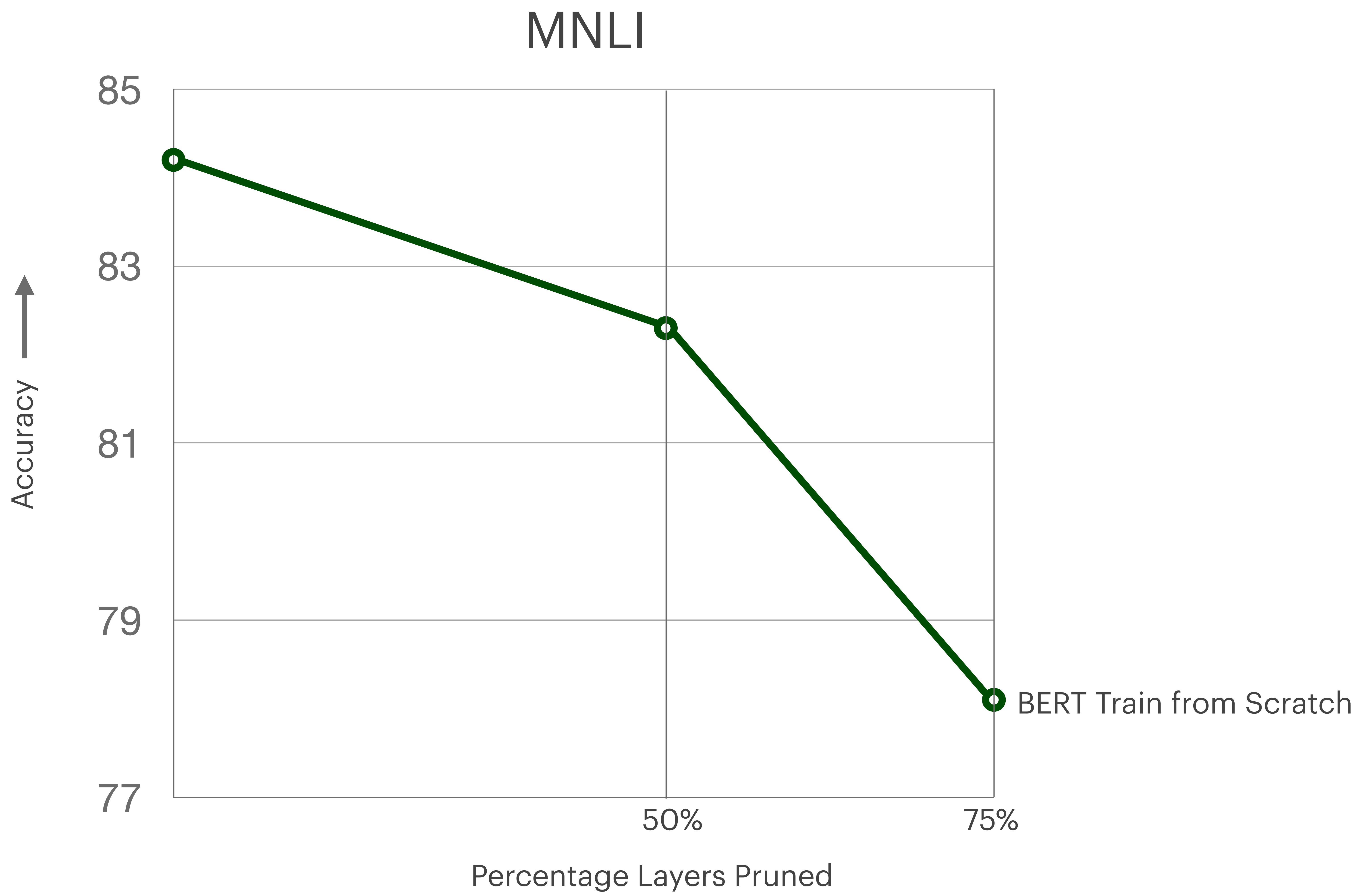
Train BERT
+ LayerDrop



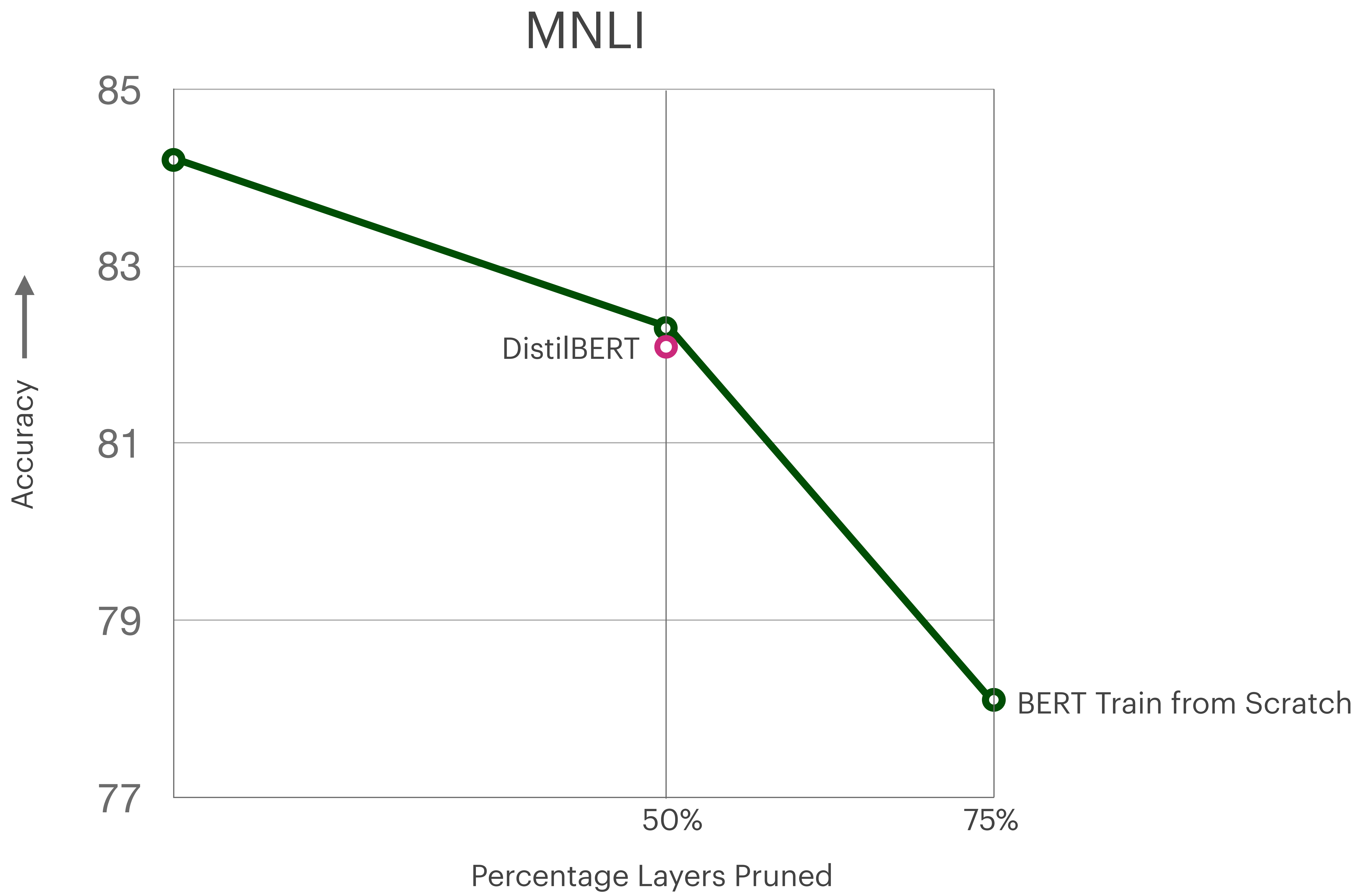
Fine-tune BERT
+ LayerDrop

PRUNE

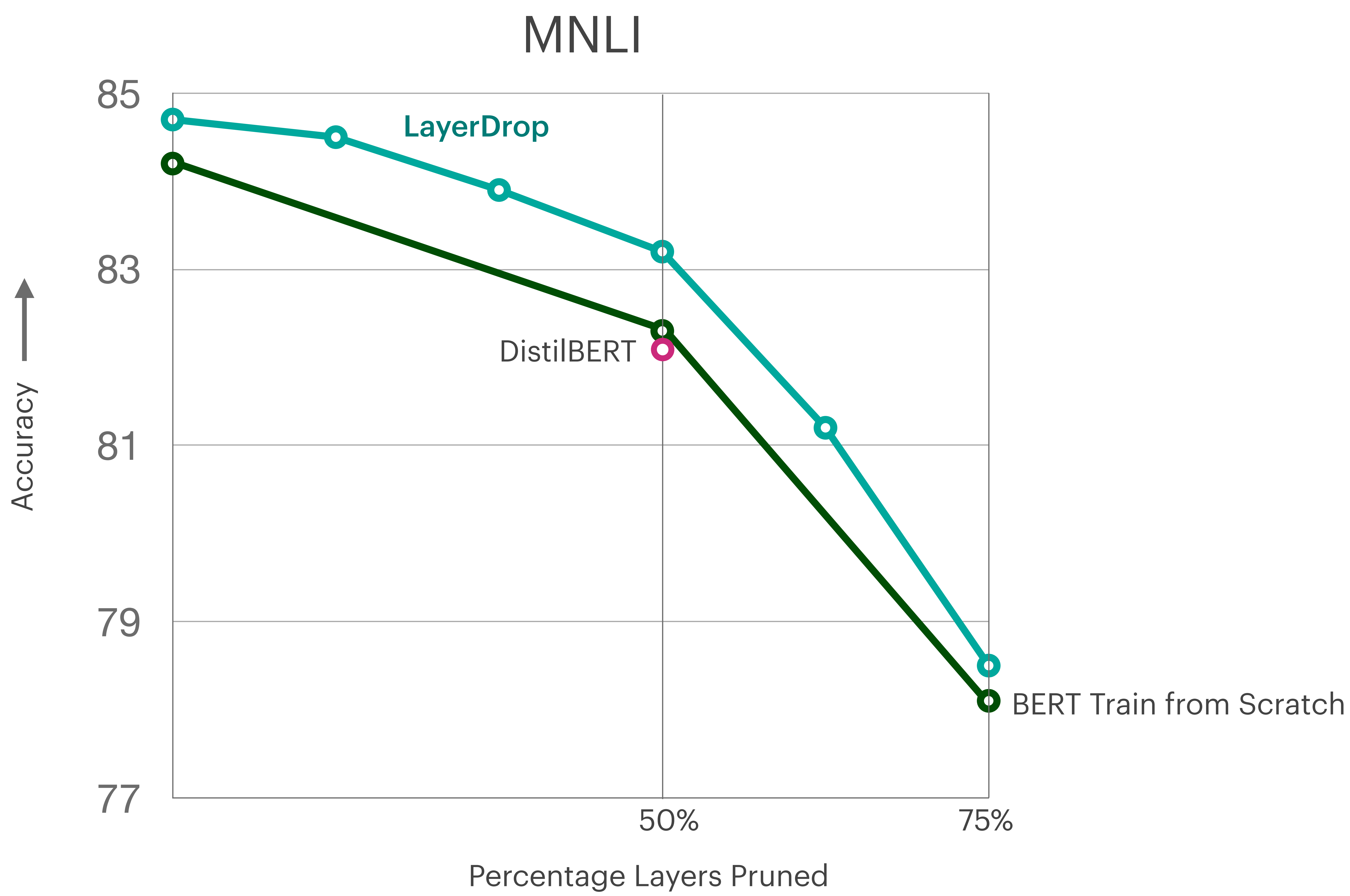
# (3) LayerDrop for Pruning - BERT Pre-Training



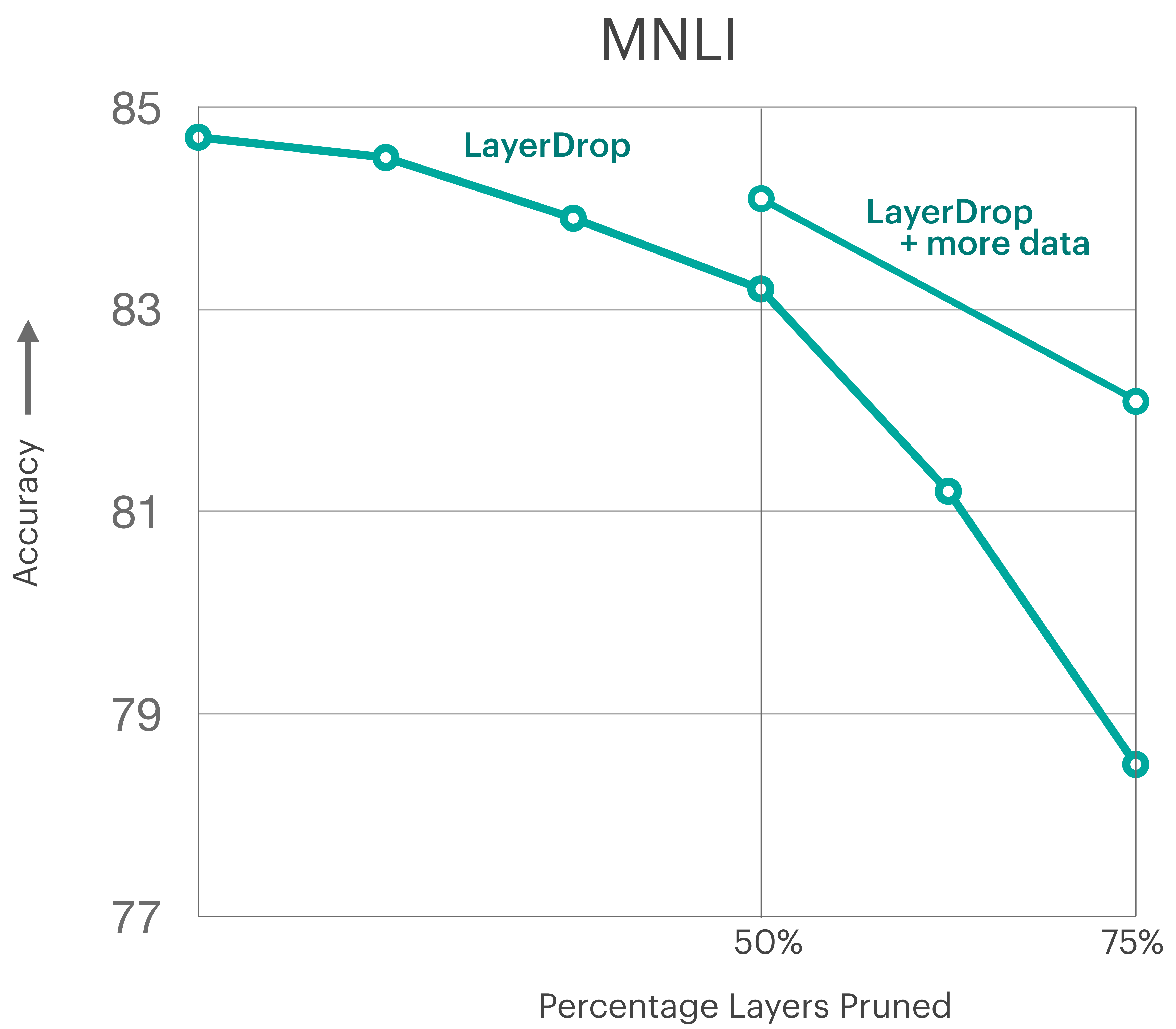
# (3) LayerDrop for Pruning - BERT Pre-Training



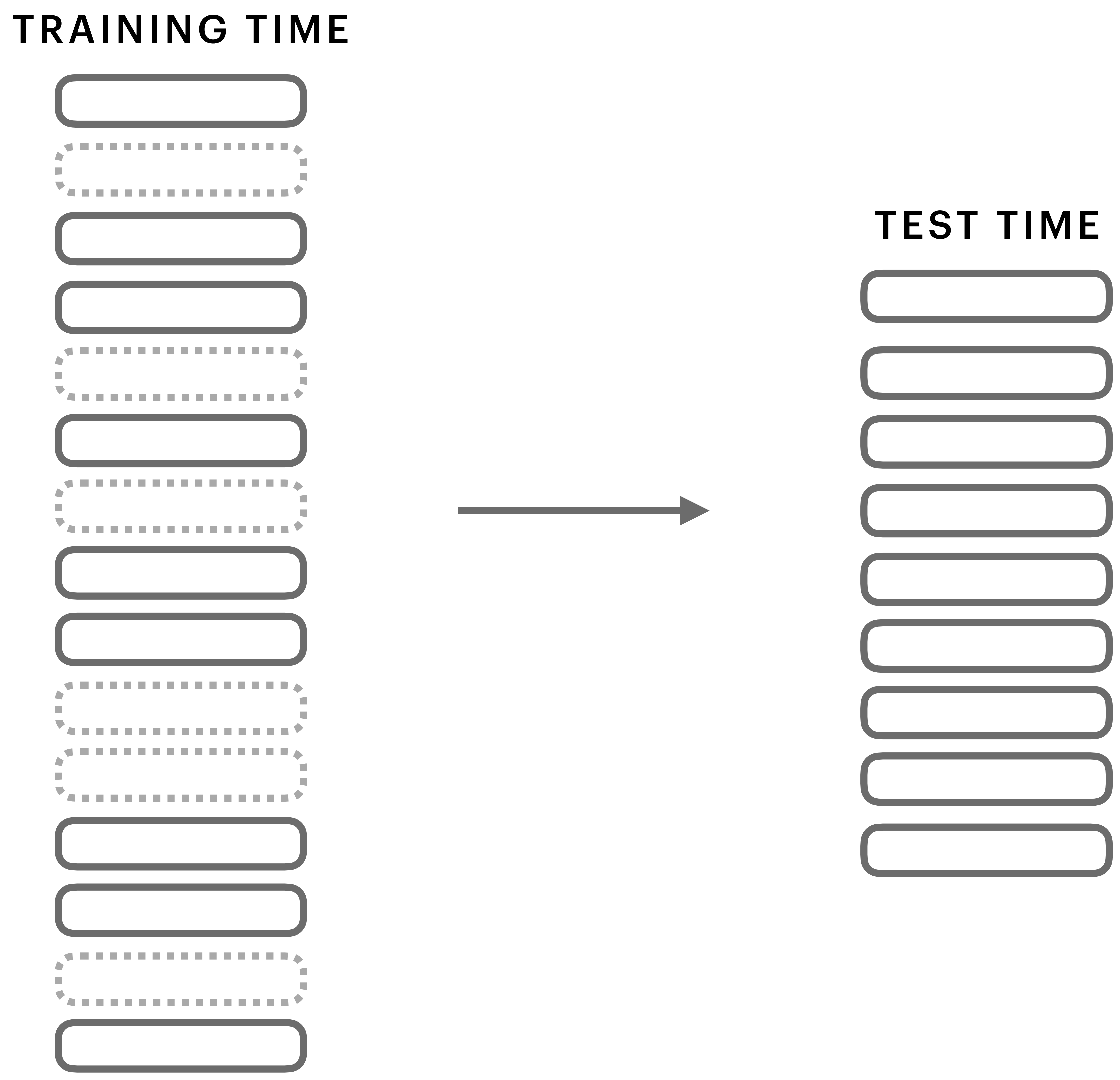
# (3) LayerDrop for Pruning - BERT Pre-Training



# (3) LayerDrop for Pruning - BERT Pre-Training



# Different Pruning Strategies - Does it Matter?



# Add LayerDrop to Your Transformer Training

```
for layer in transformer.layers:
    x = layer(x)
```

# Add LayerDrop to Your Transformer Training

```
for layer in transformer.layers:
     if random(0,1) > layer_drop and self.training:
          x = layer(x)
```

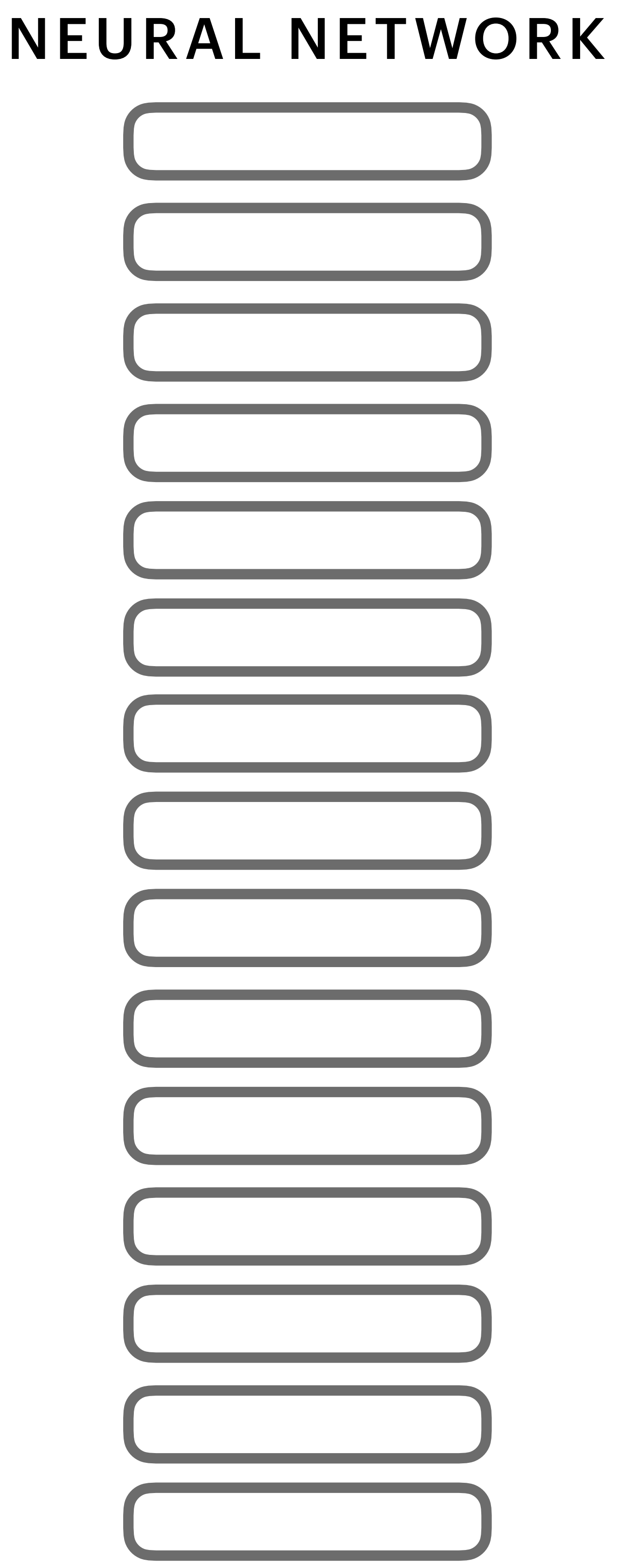
# Pruning with LayerDrop

- Training Time ✓
- Inference Time ✓
- Performance ✓
- Model Size ✓

# Techniques for Smaller Networks

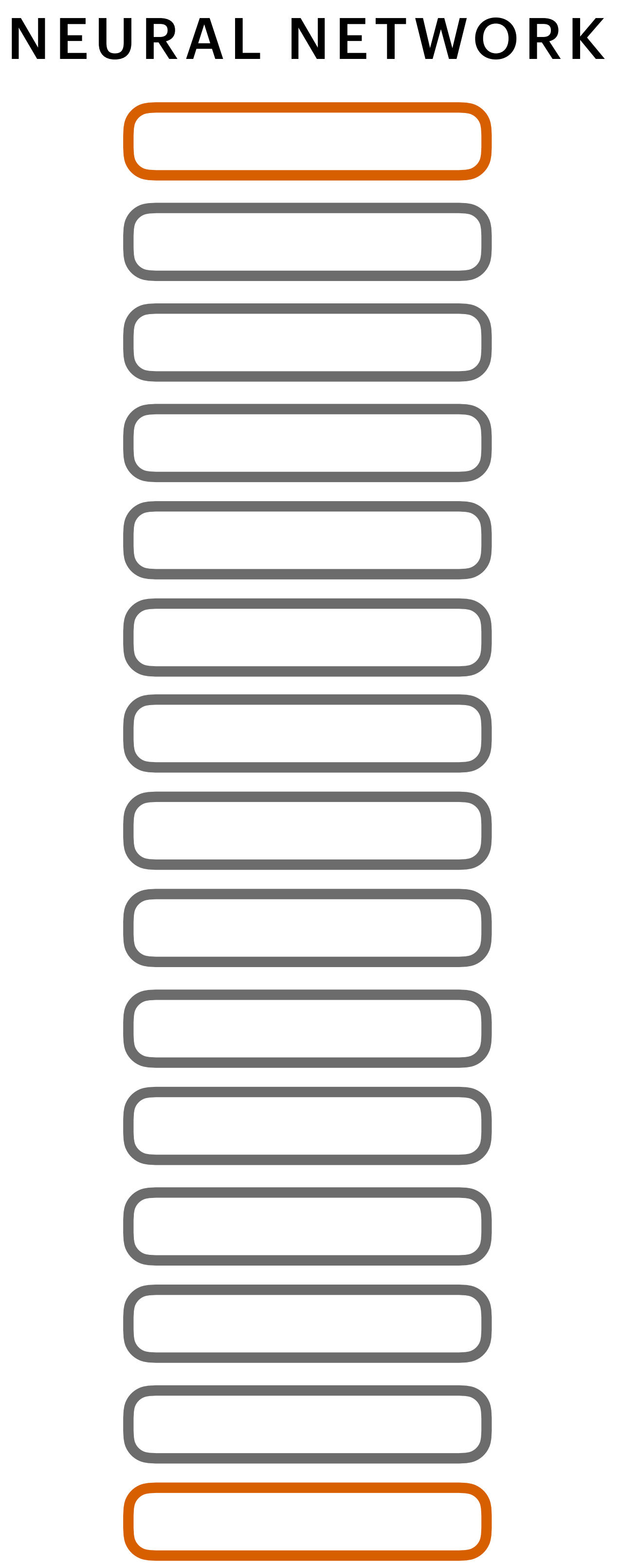
- Train Smaller Network from Scratch
- Sparsity Inducing Training
- Knowledge Distillation
- Pruning
- **Weight Sharing**
- Quantization
- More efficient architectures

# Weight Sharing




re-use weights in multiple places

# Weight Sharing

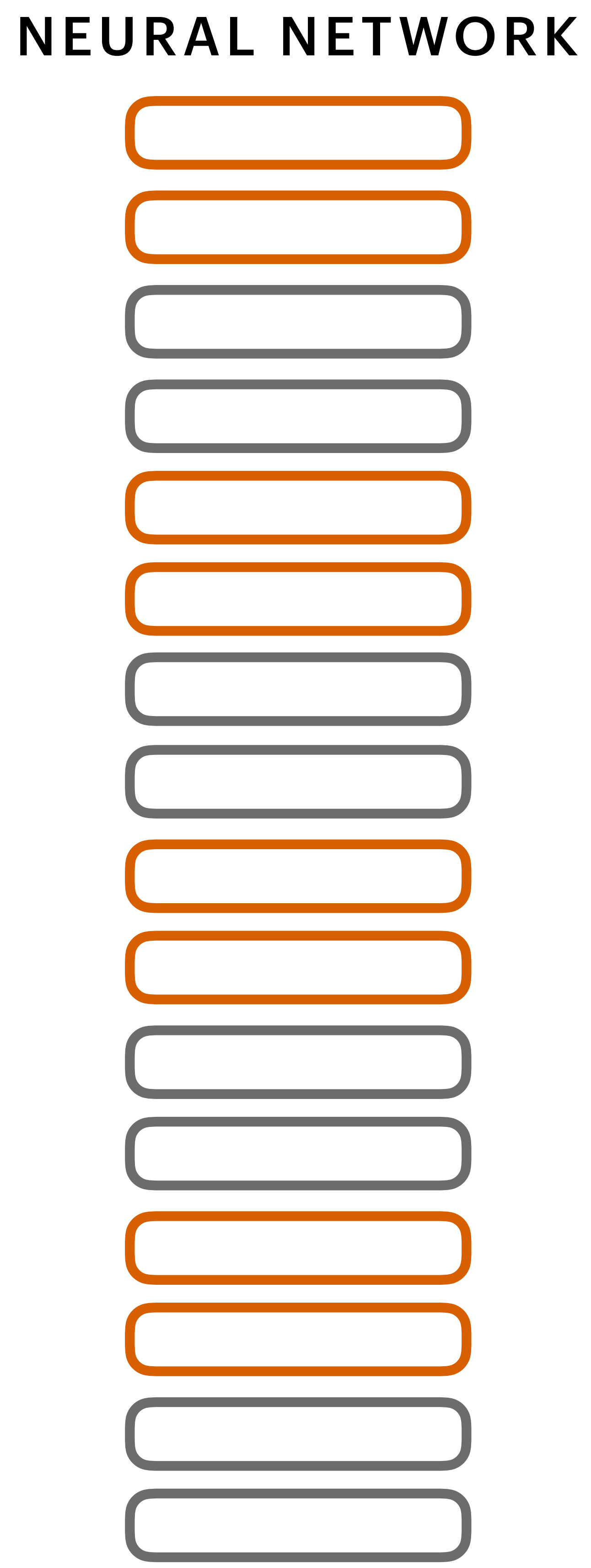



input and output embeddings tied (common)

EXTREME LANGUAGE MODEL COMPRESSION WITH OPTIMAL SUBWORDS AND SHARED PROJECTIONS

ZHAO ET AL

# Weight Sharing

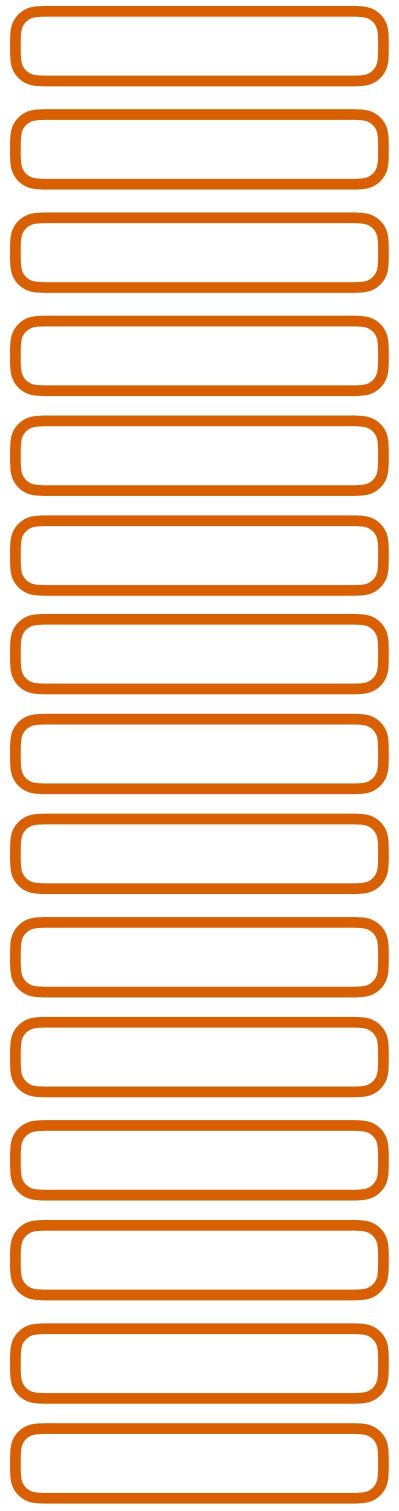



share the weights of chunks of layers

FAN ET AL

# Weight Sharing

NEURAL NETWORK



share the weights of chunks ALL layers

ALBERT: A LITE BERT FOR SELF-SUPERVISED LEARNING OF LANGUAGE REPRESENTATIONS

LAN ET AL

UNIVERSAL TRANSFORMER

DEGHANI ET AL

# Weight Sharing

- Model Size ✓ ✗
- Performance ✗ ✓

unless you increase model size

# Techniques for Smaller Networks

- Train Smaller Network from Scratch
- Sparsity Inducing Training
- Knowledge Distillation
- Pruning
- Weight Sharing
- **Quantization**
- More efficient architectures

# Advantages of Quantization

- Easily combined with existing techniques
  - you can quantize a pruned model, quantize a distilled model, etc

# Advantages of Quantization

- Easily combined with existing techniques
    - you can quantize a pruned model, quantize a distilled model, etc

- Ships with PyTorch and Tensorflow
    - easy to apply

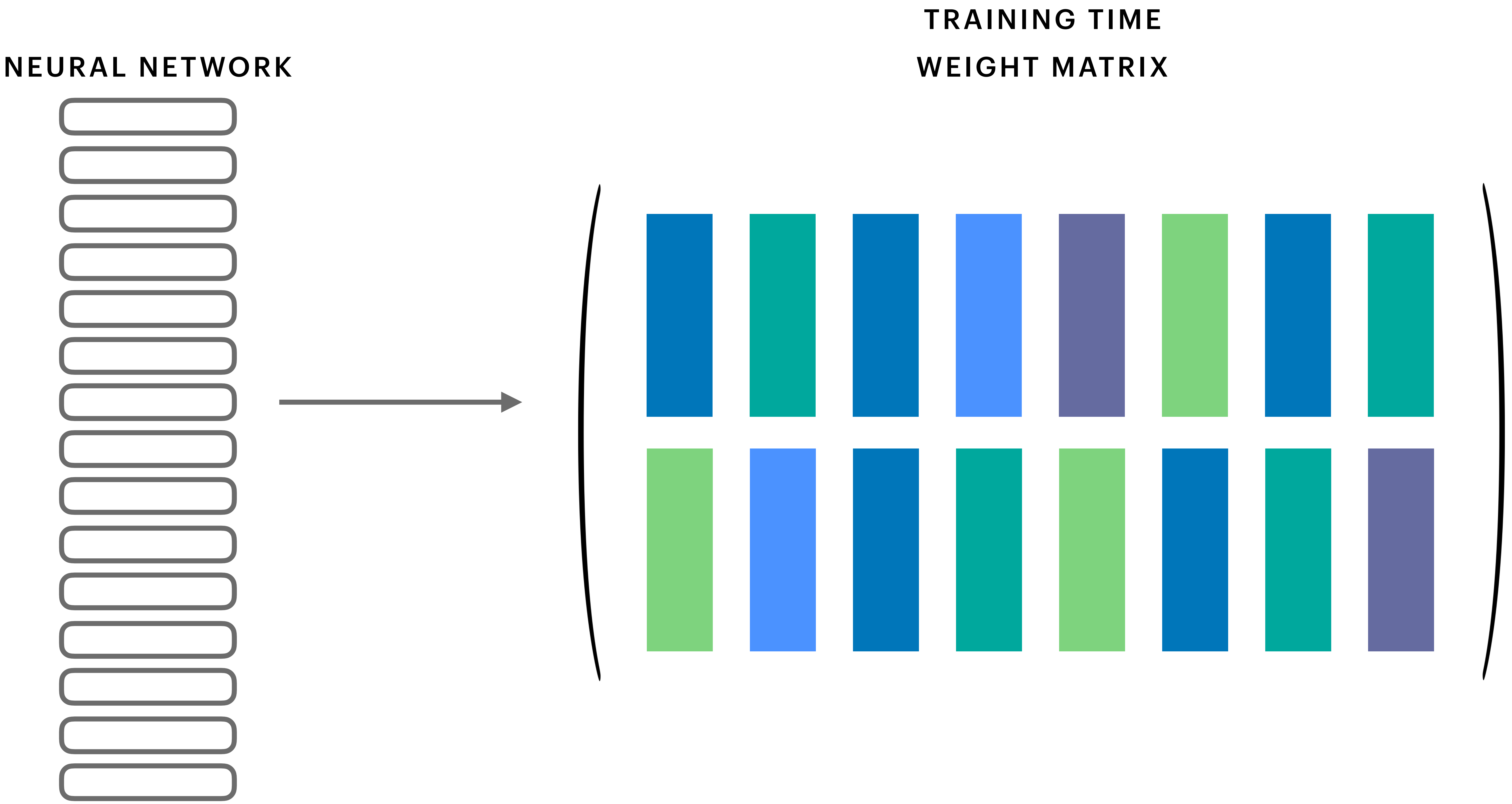
# Advantages of Quantization

- Easily combined with existing techniques
  - you can quantize a pruned model, quantize a distilled model, etc
- Ships with PyTorch and Tensorflow
  - easy to apply
- Can offer drastic compression

# How much Model Size do you want to decrease?

| | |
|---|---|
| Train Smaller Network from Scratch | maybe model will be 2-4x smaller |
| Sparsity Inducing Training | probably less... |
| Knowledge Distillation | 2-10x smaller |
| Pruning | 2x smaller |
| Weight Sharing | 2-8x smaller |
| **Quantization** | **4-25x smaller**<br>50 - 100x in combination |

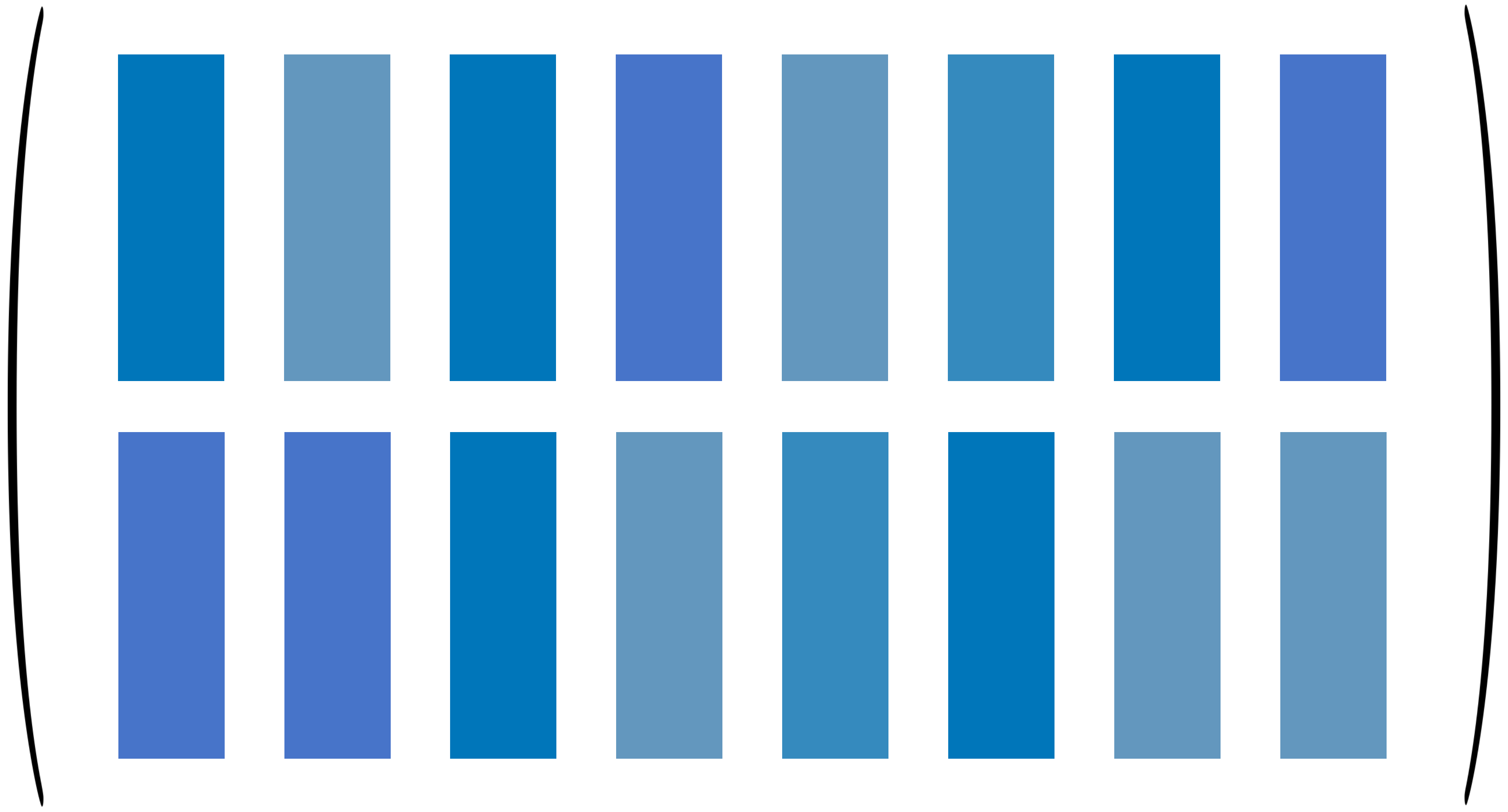
# How does Quantization offer such extreme compression?



# How does Quantization offer such extreme compression?

INFERENCE TIME

QUANTIZED WEIGHT MATRIX



QUANTIZATION OPERATIONS CHANGE THE WEIGHT VALUES
IN ORDER TO COMPRESS NETWORK SIZE

# Different Types of Quantization

- Scalar Quantization (int8, int4, binary)
- Vector Quantization (Product Quantization)

# Scalar Quantization

4x compression from int8

8x compression from int4

32x compression from binary... so far not working for Transformers

# Scalar Quantization

- Neural Networks are often stored in fp32. We save space by going to int8 or int4.
- Take real numbers and instead store them as integers with scaling factors

0.0269
real number

# Scalar Quantization

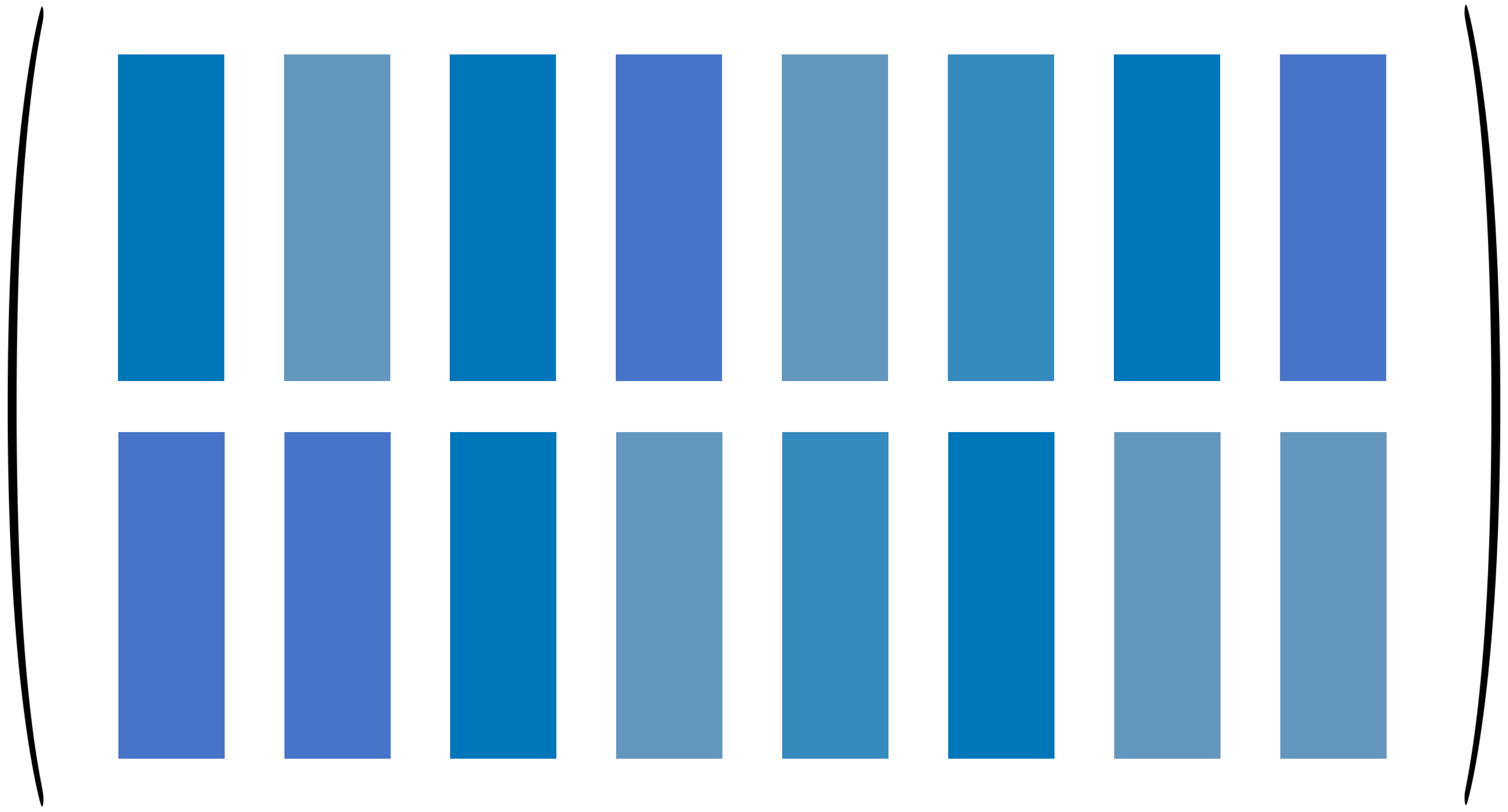
- Neural Networks are often stored in fp32. We save space by going to int8 or int4.
- Take real numbers and instead store them as integers with scaling factors

| 0.0269 | = | 110 | * | 2^ -12 |
|---|---|---|---|---|
| real number | | integer | | scaling factor |

# How to apply to an entire matrix?

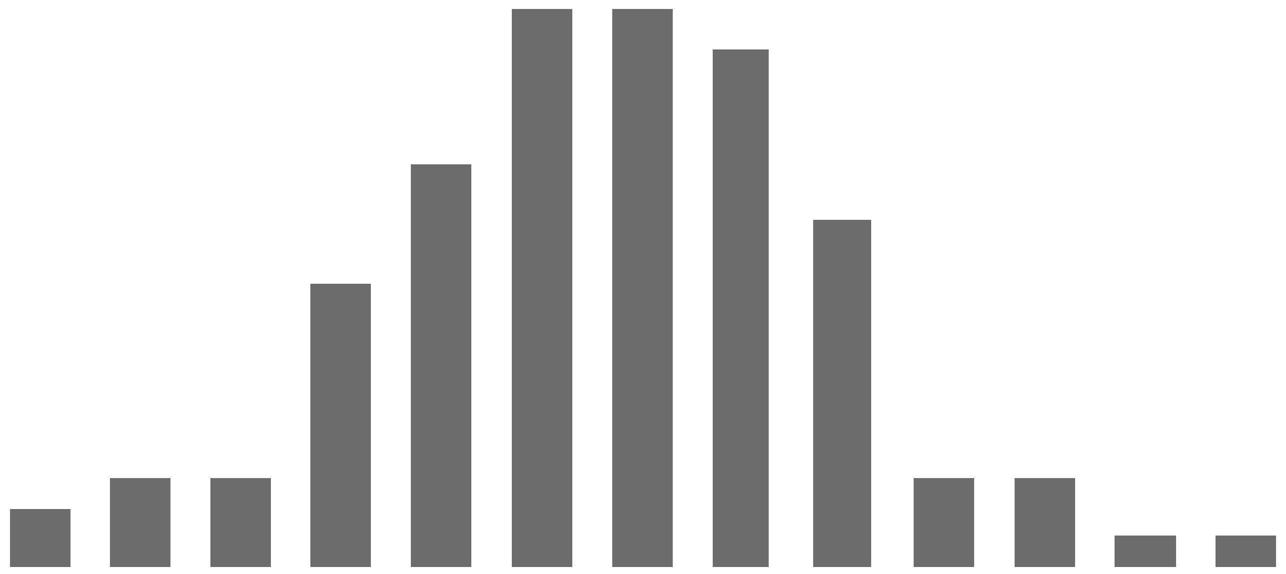
INFERENCE TIME

QUANTIZED WEIGHT MATRIX



QUANTIZATION OPERATIONS CHANGE THE WEIGHT VALUES
IN ORDER TO COMPRESS NETWORK SIZE

# Calculate Scaling Factor across all values

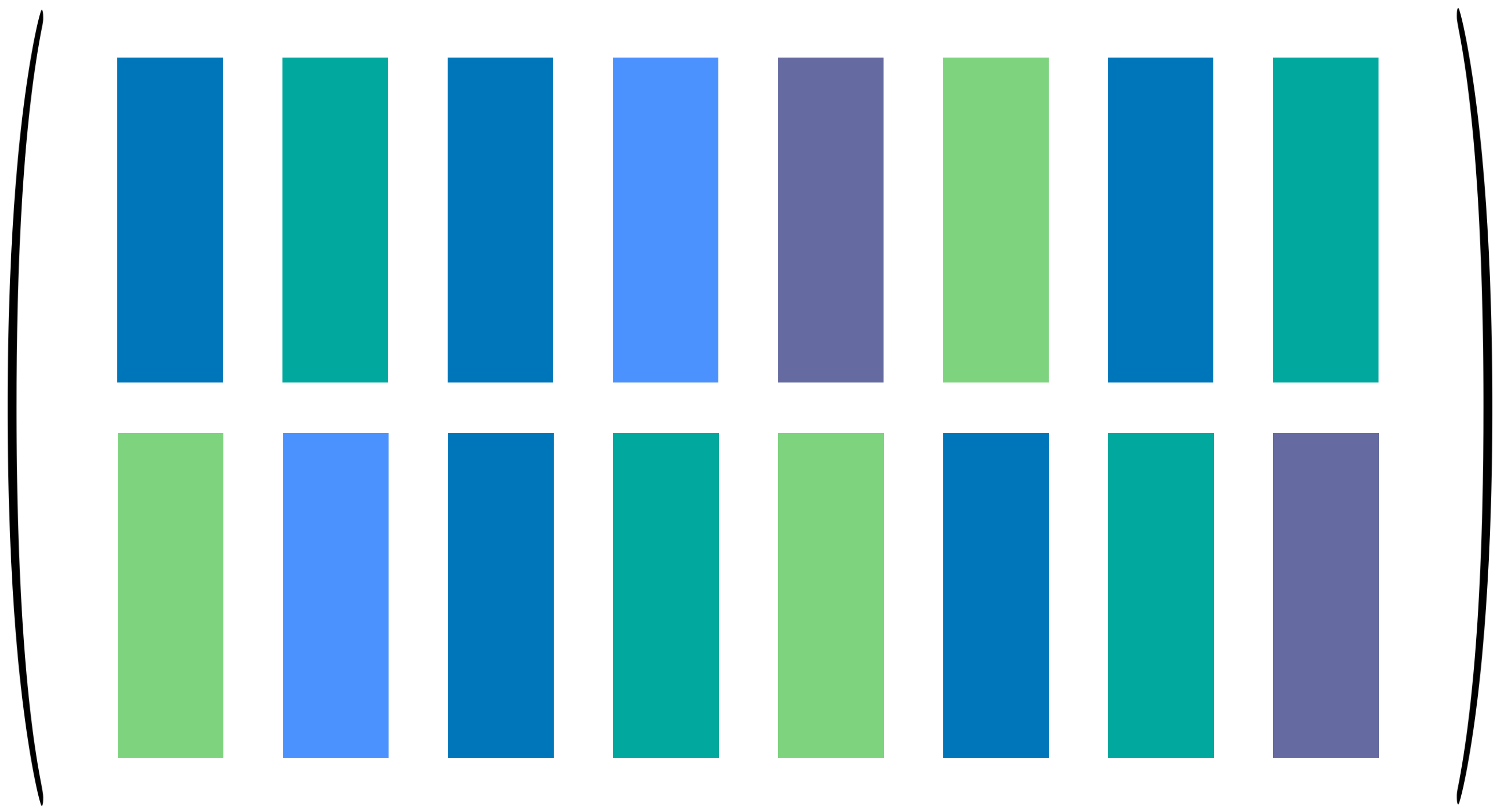


HISTOGRAM OF SCALING FACTORS

# Vector Quantization: Product Quantization

25x compression or more

TRAINING TIME
WEIGHT MATRIX



# Vector Quantization: Product Quantization

25x compression or more

Subvectors

Codewords

AND THE BIT GOES DOWN: REVISITING THE QUANTIZATION OF NEURAL NETWORKS
STOCK ET AL

# Vector Quantization: Product Quantization

can combine scalar and vector quantization

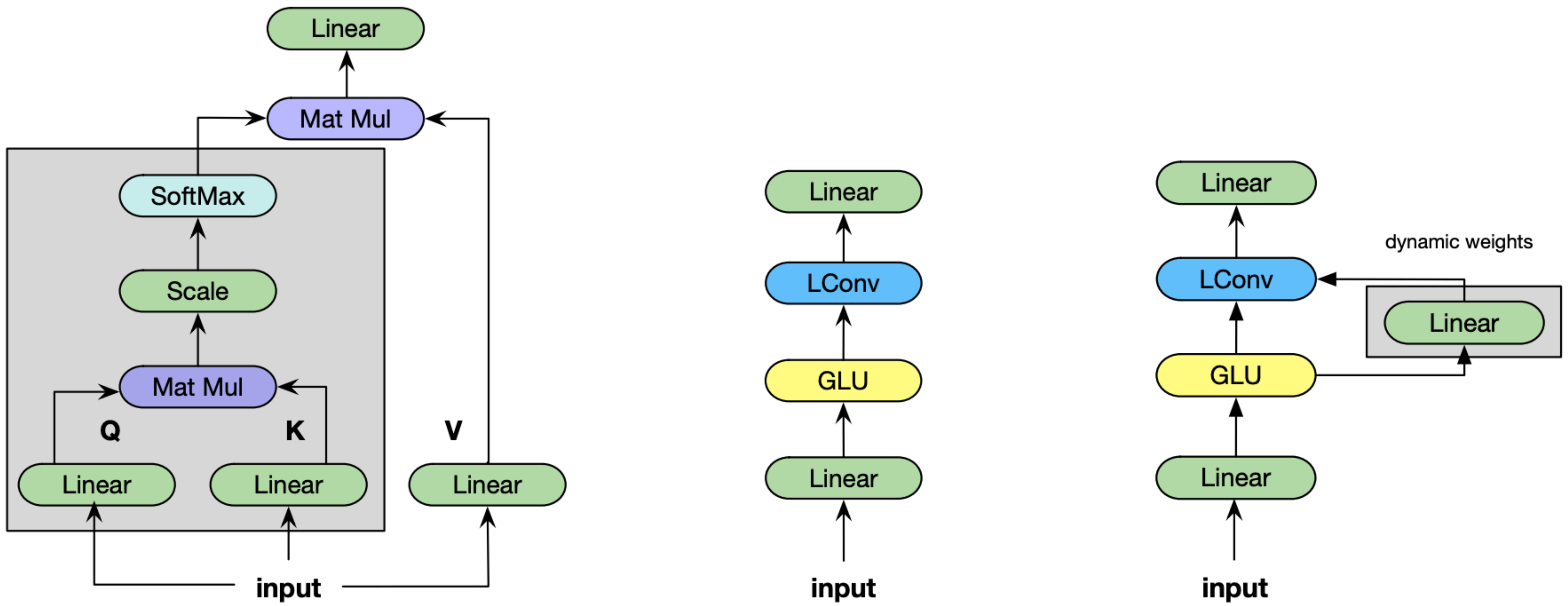
# Quantization

- Inference Time — if scalar quantization
- Performance — depending on how compressed
- Model Size — critical for on-device

# Techniques for Smaller Networks

- Train Smaller Network from Scratch
- Sparsity Inducing Training
- Knowledge Distillation
- Pruning
- Weight Sharing
- Quantization
- **More efficient architectures**

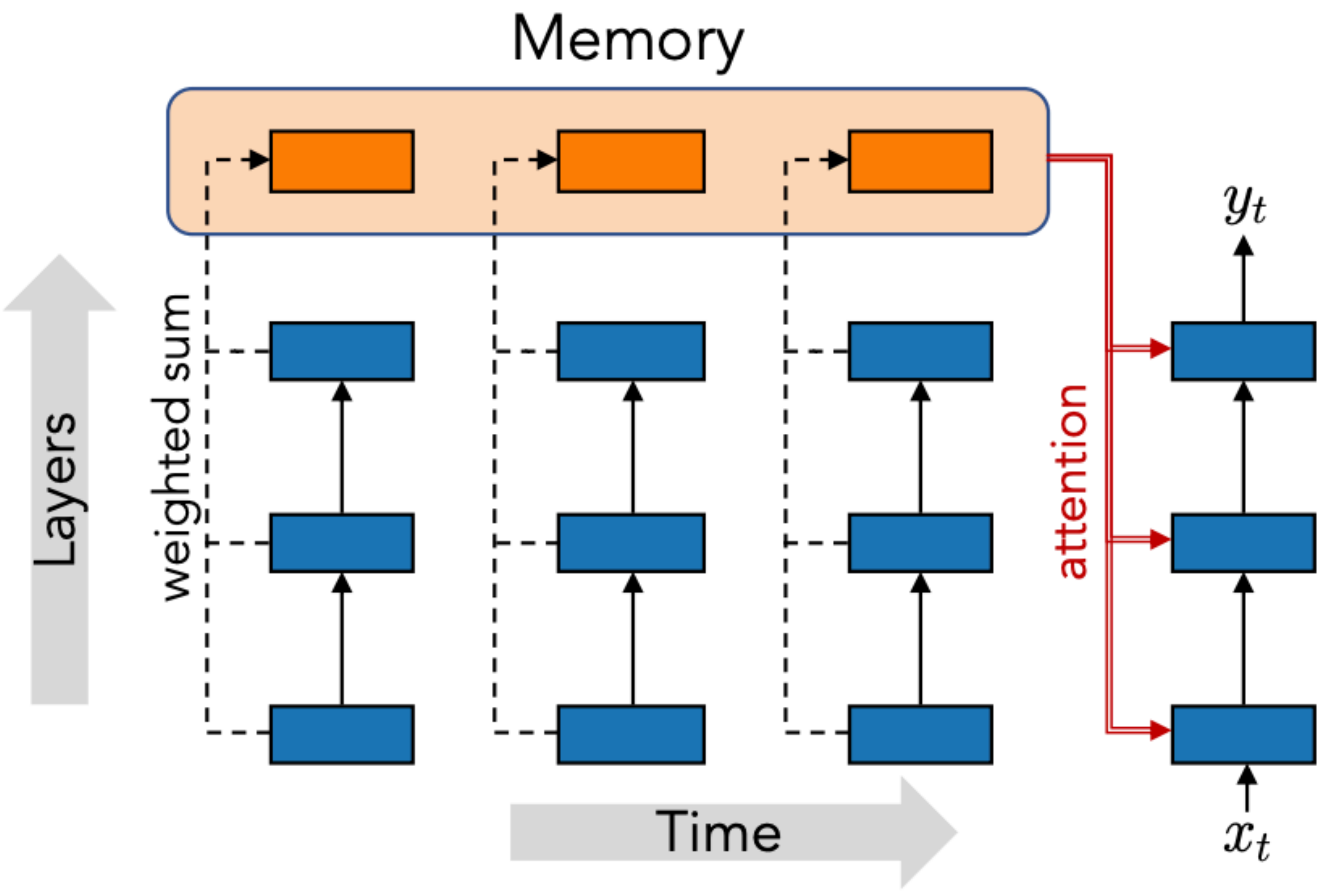
# Variant Transformer Architectures




(a) Self-attention (b) Lightweight convolution (c) Dynamic convolution

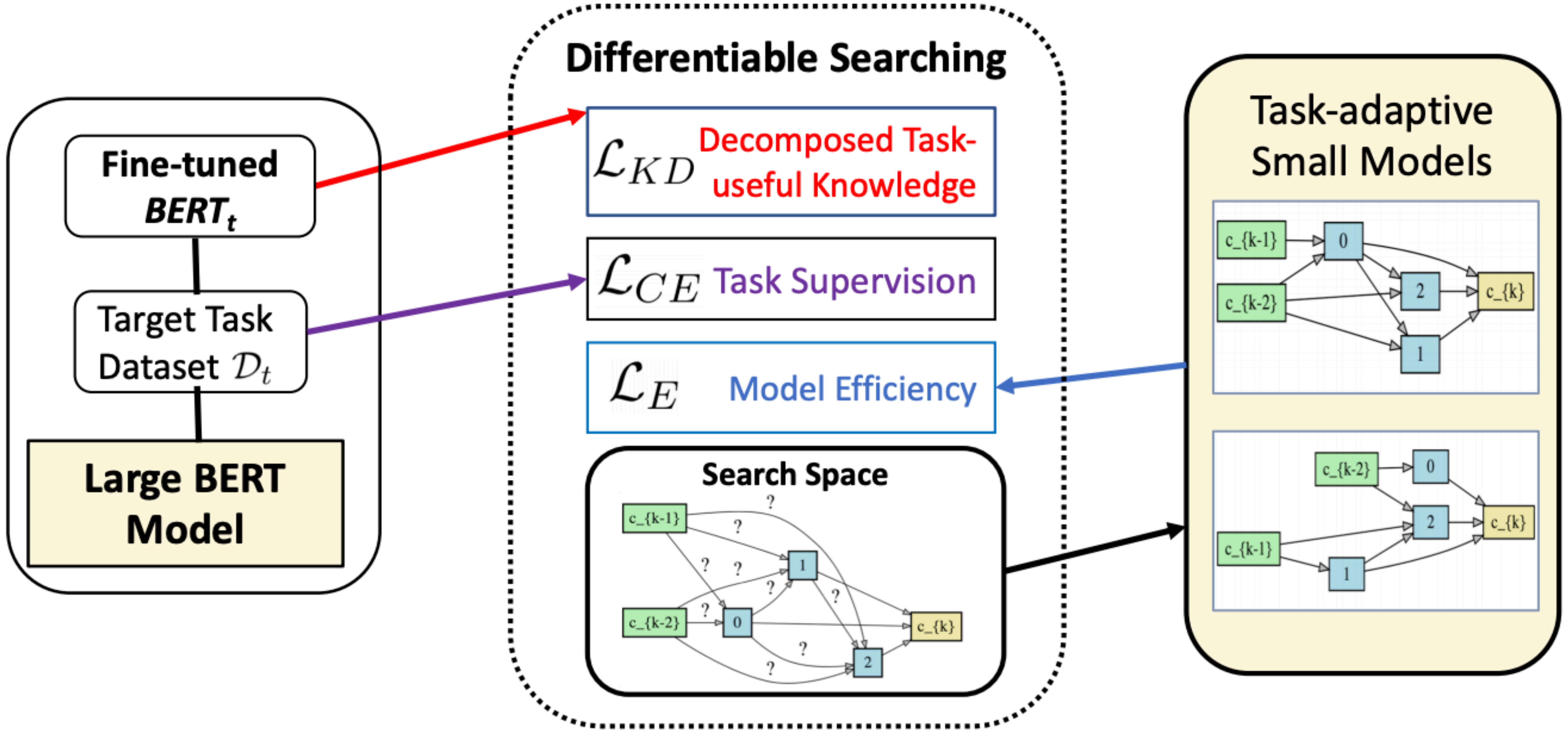
PAY LESS ATTENTION WITH LIGHTWEIGHT AND DYNAMIC CONVOLUTIONS
WU ET AL

# Variant Transformer Architectures




ACCESSING HIGHER-LEVEL REPRESENTATIONS IN SEQUENTIAL TRANSFORMERS WITH FEEDBACK MEMORY
FAN ET AL

# Variant Transformer Architectures




ADABERT: TASK-ADAPTIVE BERT COMPRESSION WITH DIFFERENTIABLE NEURAL ARCHITECTURE SEARCH

CHEN ET AL

# In Summary...

- Train Smaller Network from Scratch — strong baseline. would not discount.
- Sparsity Inducing Training
- Knowledge Distillation
- Pruning
- Weight Sharing
- Quantization
- More efficient architectures

# In Summary...

- Train Smaller Network from Scratch
- Sparsity Inducing Training
- Knowledge Distillation
- Pruning
- Weight Sharing
- Quantization
- More efficient architectures

important for on-device

# In Summary...

- Train Smaller Network from Scratch
- Sparsity Inducing Training
- Knowledge Distillation — a lot of recent improvements, very flexible
- Pruning
- Weight Sharing
- Quantization
- More efficient architectures

# In Summary...

- Train Smaller Network from Scratch
- Sparsity Inducing Training
- Knowledge Distillation
- Pruning
- Weight Sharing
- Quantization
- More efficient architectures

easy and straightforward gains

# In Summary...

- Train Smaller Network from Scratch
- Sparsity Inducing Training
- Knowledge Distillation
- Pruning
- Weight Sharing
- Quantization
- More efficient architectures

depends on how much you share

# In Summary...

- Train Smaller Network from Scratch
- Sparsity Inducing Training
- Knowledge Distillation
- Pruning
- Weight Sharing
- Quantization
- More efficient architectures

important for on-device. easily combinable.
can be used for aggressive compression

# In Summary...

- Train Smaller Network from Scratch
- Sparsity Inducing Training
- Knowledge Distillation
- Pruning
- Weight Sharing
- Quantization
- More efficient architectures

exciting direction

# And of course, even more considerations…

- Latency (faster decoding)
- Models that fit on Specialized Hardware
  - specific block sizes
  - battery life and heat from device

Interested in Efficient NLP?

Simple and Efficient Natural Language Processing

Workshop at EMNLP 2020 in Punta Cana

Thanks for listening!